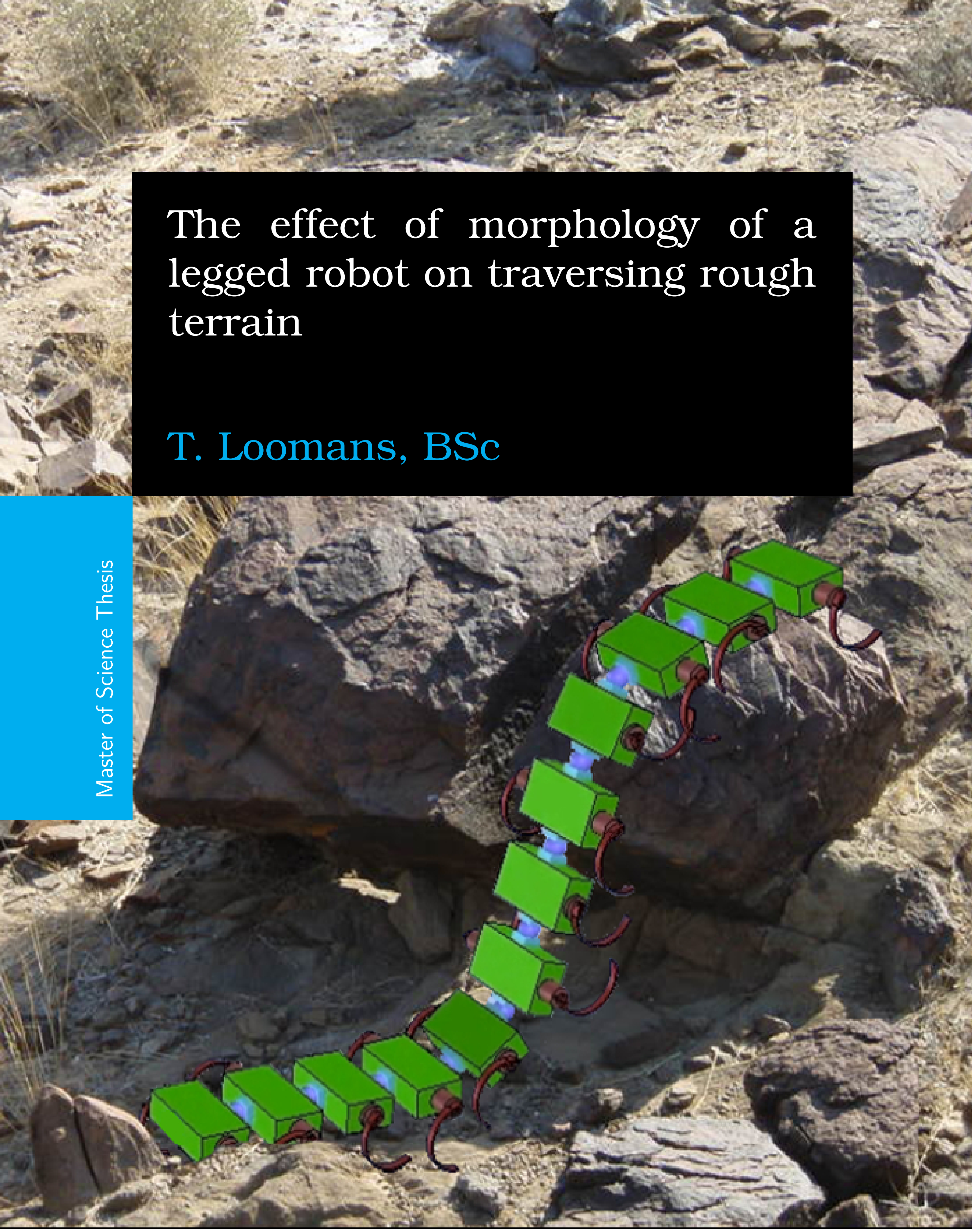
The effect of morphology of a legged robot on traversing rough terrain

T. Loomans, BSc

Master of Science Thesis



Delft University of Technology

Delft Center for Systems and Control

The effect of morphology of a legged robot on traversing rough terrain

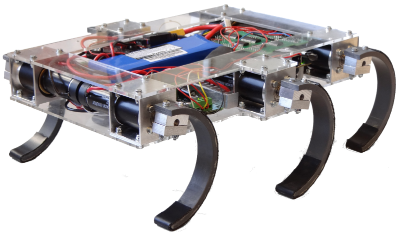
Master of Science Thesis

For the degree of Master of Science in BioMechanical Engineering at
Delft University of Technology

T. Loomans, BSc

October 13, 2015

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology



With special thanks to the TU Delft zebro and cyberzoo team.

Abstract

When a disaster strikes a human-engineered environment, it can be difficult and dangerous for rescue workers to search for survivors. Collapsed buildings, fire and other hazards can make specific areas too dangerous to send in a human or dog. In this case, urban search and rescue (USAR) robots can be useful. Various researches have been conducted on small legged robots navigating over rough terrain. The design of these robots varies amongst researches, ranging from short four-legged robots to long flexible centipede-like models and even modular robots. Often, a new body morphology feature is tested by incorporating it in a new design and testing the performance of the robot on a rough terrain landscape. However, in this way the new feature is tested but not compared against other body morphologies on the same landscape. Therefore it is difficult to say which design is the most promising. In this thesis project, different body morphologies are tested and compared in simulation on various obstacles. A model of a modular centipede robot was designed, consisting of small bipedal units that can connect together and form a centipede of various lengths. Extensive simulations were conducted to be able to compare different connection types between the units (one or two active/passive degrees-of-freedom), and different robot lengths (changing the number of connected units). The results show that on a fractal landscape, adding degrees-of-freedom to the mechanical design of a modular centipede robot will not result in a better performance. Also, increasing the length shows only little improvement if the robots consists of more than 5 units. For a step or gap obstacle, an actively controlled pitch motion must be incorporated in the spine, preferably with a high torque. Increasing the length of a centipede robot with an actively controlled spine will enable it to cross a larger gap or climb a higher step. This thesis project has laid the ground work for the design of a new urban search and rescue robot using modular bipedal units. The results may be used as a guide for the mechanical design of such a robot.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Dr. G.A.D. Lopes for his assistance during the writing of this thesis.

Delft, University of Technology T. Loomans, BSc
October 13, 2015

“To improve is to change; to be perfect is to change often.”
— *Winston Churchill*

Part I

Preliminary Knowledge

Chapter 1

Introduction

1-1 Background

When a natural disaster strikes a populated area, it can be devastating to its inhabitants. Buildings get destroyed, leaving a terrain filled with debris. Often survivors are trapped beneath piles of rubble and are in need of rescue. It is of vital importance that they are located as fast as possible so that rescue workers can devise a plan to free them from their plight and give them medical attention. Unfortunately, disaster stricken areas are difficult to navigate safely, retarding the rescue process and decreasing the survival chance of the victims. Nowadays rescue teams deploy dogs to search for victims, but they are in danger of getting themselves trapped or injured as well. Therefore the use of Urban Search and Rescue (USAR) robots is favourable. USAR robots are designed to navigate through rough terrain and locate victims with the use of a wide range of sensors, including Laser Imaging Detection And Ranging (LIDAR) and thermal cameras. Ideally, they can be deployed in large numbers to cover a disaster area rapidly and report their findings to the rescue team. Nowadays USAR robots are not yet widely used since the current models are not reliable enough and cannot conquer every obstacle found in disaster areas. To be useful in critical search and rescue situations, a robot must be able to:

- Move over rubble and debris
- Crawl in tight spaces
- Search for victims autonomously
- Carry payloads such as sensors, batteries and equipment to aid victims
- Stay connected with the rescue workers and provide them with information

Robocup Rescue In order to “ increase awareness of the challenges involved in search and rescue applications, provide objective evaluation of robotic implementations in representative environments, and promote collaboration between researchers ”, the Robocup Rescue Project [1] was started after the Great Hanshi-Awaji earthquake which hit Kobe City on the 17th of January 1995. In this yearly championship, different universities from all over the world compete with their robots in simulated disaster areas. The rescue arenas constructed to host these competitions are based on the Reference Test Arenas for Urban Search and Rescue Robots developed by the U.S. National Institute of Standards and Technology (NIST). There are three different arenas with increasing difficulty levels. They consist of a maze of walls, doors, elevated floors and other physical obstacles. The objective of the robots is to map the area and find victims. The simulated victims consist of clothed mannequins (both adult and child size) which emit body heat, make motions, produce sounds, and emit carbon dioxide to simulate breathing. The victims are distributed throughout the environment in roughly the same situational percentages found in actual earthquake statistics. The robots can earn points by finding victims and reporting their location and state on a human readable map. Since the test arenas closely resemble real disasters areas, a robot that performs well in the competition might also be useful for rescue workers in real rescue situations.

Most of the robots competing in the Robocup Rescue competition use caterpillar tracks and are large and heavy. This design enables them to overcome large obstacles and rough terrain quickly, but in a real rescue mission it would be dangerous to drive such a large and heavy machine near victims. Perhaps a lighter legged design would be more practical for a real-life rescue situation. The Delft University of Technology (TU Delft) has been working on a small lightweight legged rough terrain robot called Zebro, which will be introduced in section 1-3. This thesis project focusses on the next design steps for this robot. In order to increase the rough terrain capabilities and investigate possible future upgrades in the design, various potent morphologies found in other rough terrain robots are simulated on obstacles found in search and rescue situations. The results are used to investigate whether certain adjustments (changing the shape or adding passive or active degrees-of-freedom (DOFs)) increase the competence of a Zebro-like design for a USAR robot.

1-2 Challenges for USAR robots

For a robot to be able to navigate through a disaster area and search for victims, it has to deal with a range of different challenges. The debris of collapsed buildings creates an uneven terrain with gaps, steps, tight spaces, water, fire, etc. The goal is to find all victims as fast as possible and report back their location and condition to the rescue workers, along with information about the environment. Since a disaster area can be large, it is better to think of a complete robotic platform than just a single robot. It is not necessary, and not very practical, for a single robot to be specialized in all the tasks that are needed during a rescue operation. A rescue team could consist of human rescue workers working together with different types of robots. Unmanned Areal Vehicles (AUVs) can scout the area from the air, providing information to robots moving over the terrain. The different robots could also help each other to overcome obstacles. Think for instance of flying robots picking up and transporting ground robots, or multiple ground robots pushing or pulling each other over obstacles. To narrow down the design space, this thesis project will only cover ground robots navigating over rough terrain. There are many challenges in designing such a robotic system, for instance how can it cover a large area efficiently, what should the robots look like, how will they help each other overcoming obstacles and what type of obstacles should they be able to overcome?

1-3 State of the Art in USAR Robots

Today, USAR robots are not broadly used by rescue teams (with some exceptions of specialized USAR robots teams). However, a few models are already in use, mainly for military purposes. The three most capable rough terrain robots in use today will be reviewed in this section.

One of the robots used in military missions is Packbot, which is produced by a company called iRobot. This robot runs on two caterpillar tracks and has two extra caterpillar tracks on a pivoting arm in its front. These are used to climb over obstacles larger than the height of the robot by placing the arms on top of the obstacle and pushing down to create traction so the robot can pull itself over the obstacle. The Packbot robot has a robotic arm attached to its top that can be remotely controlled. A range of tools can be attached to this arm so the robot can be adjusted to various situations. These attachments range from different types of sensors and cameras to grippers and cutting tools. The Packbot robot can be useful for remote bomb disposal or inspection of dangerous areas, however it is not able to overcome all the obstacles found in disaster areas. This means that for search and rescue purposes its practicality is limited.

A well-known company involved in designing rough terrain mobile robots for military purposes is Boston Dynamics. This is an engineering and robotics company that specializes in rough terrain robots and is funded by the Defense Advanced Research Projects Agency (DARPA). They claim that:

> “Their mission at Boston Dynamics is to develop a new breed of rough-terrain robots that capture the mobility, autonomy and speed of living creatures. Such robots will travel in outdoor terrain that is too steep, rutted, rocky, wet, muddy, and snowy for conventional vehicles. They will travel in cities and in our homes, doing chores and providing care, where steps, stairways and household clutter limit the utility of wheeled vehicles. Robots meeting these goals will have terrain sensors, sophisticated computing and power systems, advanced actuators and dynamic controls.”

Their most distinguished robot is BigDog, a large quadruped robot built for rough terrain navigation which can carry payloads of more than 50kg (and even up to 150kg on flat terrain) [2]. It was designed to aid military personnel by carrying heavy equipment. This robot can climb over rough terrain by carefully placing its legs on the rubble and balancing its body. Although BigDog looks impressive in the videos available on the internet, it is not very practical for search and rescue operations. This is mainly due to its size and cost. The robot is large and complicated and moves relatively slowly at 6km/h over rough terrain. For search and rescue missions it is important to cover an area fast. A swarm of small (and preferably cheap) robots is favoured in such a situation since they can spread over the area and look for victims as a group, cutting back on the time needed to find all victims.

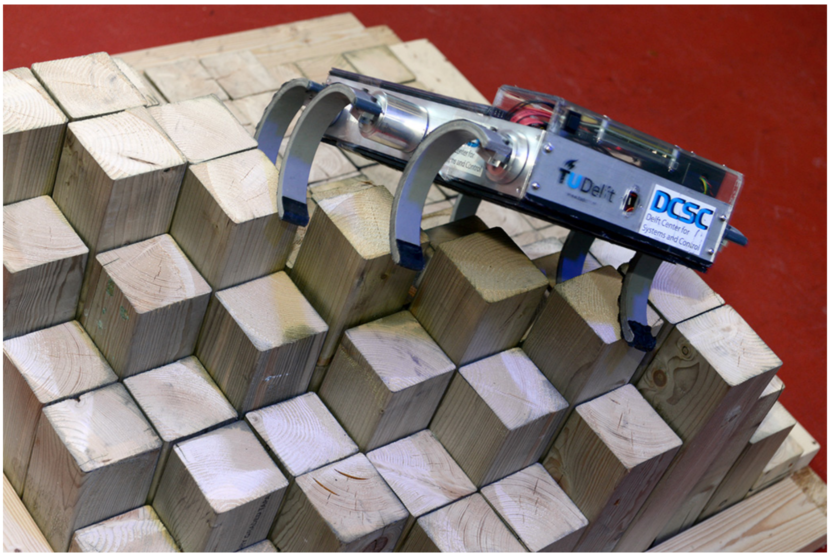


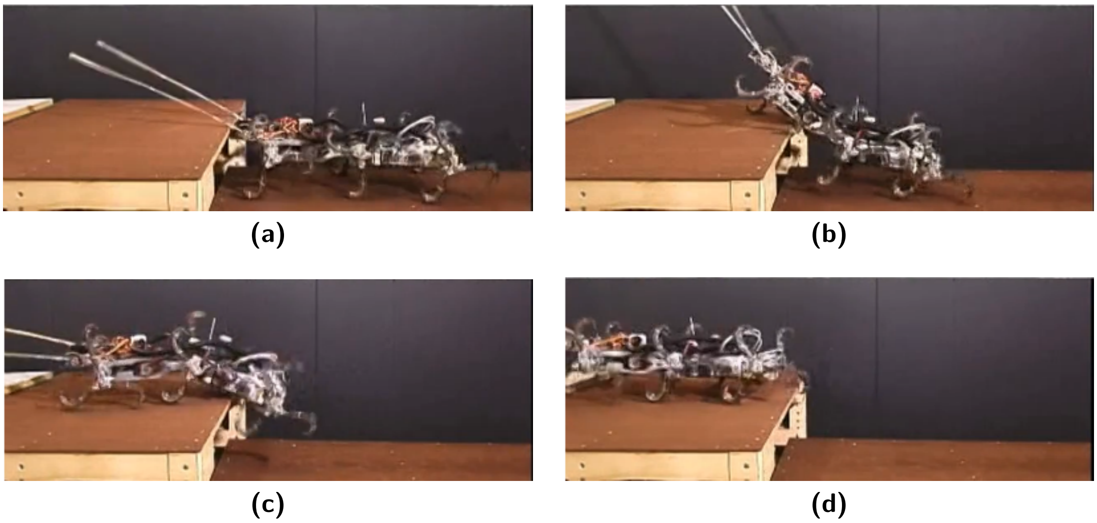
Figure 1-1: ZEBRO robot from the TU Delft.

An effort on building a fast and simple rough terrain robotic platform was made by four collaborating universities in the United States with the RHex project. RHex is a simple autonomous rough terrain hexapod robot. The project started in 1998 and was funded partly by DARPA [3]. During the project, seven different versions of the robot have been developed and tested and there are over 30 papers written about the different iterations of this design and their abilities, including various control strategies [4] [5] [6] [7] [8] [9]. The strength of the RHex design, which is based on the movement of a cockroach, is that the robot is relatively basic with only one actuator per leg. This makes the robot robust and simplifies the control. The RHex robots are some of the most researched rough terrain robots and they show a promising performance. Since they are small and can be fabricated inexpensively, they seem useful as USAR robots. This is why the RHex model will be the basis for the design of the robotic search and rescue solution in this thesis project. There is also a version of RHex available on the Delft University of Technology (TU Delft) which is called Zebro (zes-benige robot; Dutch translation)(fig. 1-1). Today still new mechanical improvements as well as updated control strategies are being developed for this robotic platform.

1-4 Interesting Robot Morphologies

Over 50 studies have been conducted on rough terrain robots, testing different body morphologies, propulsion mechanisms, and control strategies. The morphologies used in these studies are diverse and include small insect-like robots, long snake or centipede robots, and large quadrupeds. The bodies of these robots range from stiff blocks to flexible spine connections (with both active and passive joints), and also include modular constructions. The propulsion mechanisms include wheels, caterpillar tracks and legs. In the literature study preceding this thesis project, various morphologies were examined to see which are the most promising for a new USAR robotic system. In this section, some of the most promising morphologies and their advantages and disadvantages will be discussed, resulting in a theoretical robot design that will be tested in simulation.

In the early days of rough terrain robot studies, often a large effort was spend on designing the legs (or wheels) which are then attached to a simple rigid box functioning as the body. The only function of this body is to house the electronics, motors, batteries, etc, and provide a rigid connection between the leg actuators. When looking at nature it is seen that animals who navigate over rough and irregular terrain often have flexible bodies actuated by muscles, which they use to improve their speed and stability. Biomimetics is the study of the structure and function of biological systems as models for the design and engineering of materials and machines. When searching for ideas to improve movement over rough terrain, a logical step is to look at animals that do this well. Insects like cockroaches are often used as a source of inspiration, since they are relatively simple in body design and they can move over high obstacles compared to their body size. When looking at larger animals, many mammals have great rough terrain capabilities as well. Think for instance of cheetahs, goats, monkeys, cats, etc. This is why over a dozen of recent legged robots are inspired by these animals. It has been shown that incorporating a flexible spine in quadrupedal robots, enabling the body to make a pitch motion, can improve their stability and running speed [10] [11] [12] [13]. These are however experimental set-ups where the robots are running on flat ground or thread mills and not on rough terrain. The Whegs II robot [14] is an example of a rough terrain robot that uses its actuated spine to overcome obstacles. By actuating the joint that connects the two body parts of the robot, it can pitch up its front together with its front legs to reach the top of obstacles which would normally be too high to climb (fig. 1-2). Using this technique it can climb steps of 175% its leg height.


(a)

(b)

(c)

(d)

Figure 1-2: Whegs II climbing over an obstacle by lifting up its front legs.

A roll motion in the body can also be beneficial for moving over rough terrain since it disconnects wheel/leg pairs. Consider a robot with four wheels/legs and a rigid body. When it moves over an obstacle and the front right wheel is on that obstacle, for some time only the front right and rear left wheel are in contact with the terrain. If there is a joint in between the wheel pairs which enables a roll motion, all the wheels are in contact with the terrain

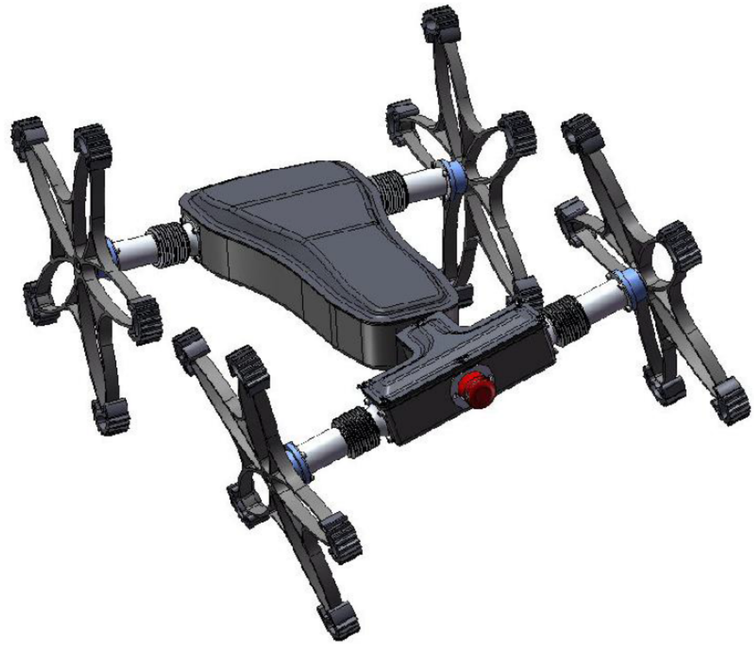
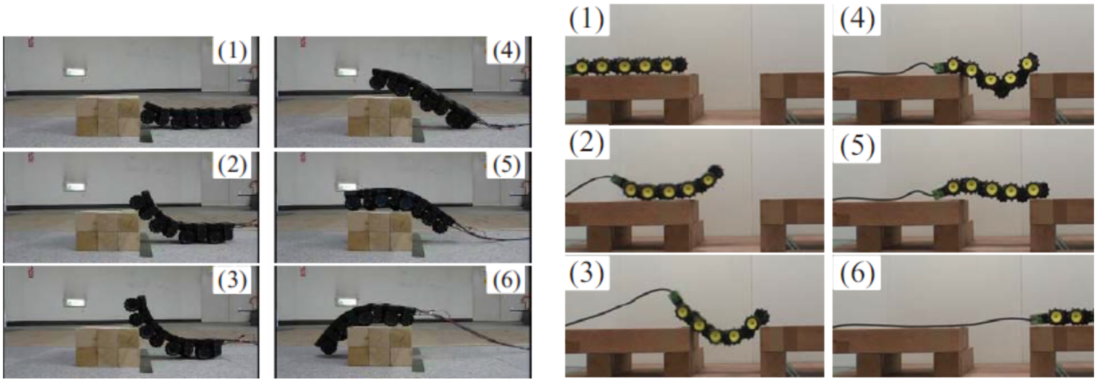


Figure 1-3: CAD model of the Asguard II robot from DFKI.

providing more traction. An example of a robot using this mechanism is the Asguard II from DFKI [15] (fig. 1-3).

Over half a dozen studies have been conducted on longer snake-like robots moving over rough terrain [16] [17] [18]. Because of the long flexible bodies with up to 24 legs, these robots still keep ground contact even on uneven terrain. This enables them to create traction and move over terrain that a rigid-bodied robot of a similar size cannot conquer. An extensive study on these types of snake-like rough terrain robots was conducted by Suzuki et al. [19] at the Tokyo Institute of Technology (Tokio Tech). The research team designed a series of robots named SOURYU-VII, SOURYU-VIII and SOURYU-IX, which were tested on rough terrain and obstacles like steps and gaps. With their actuated flexible spine they are able to climb steps up to three times their body height (with a maximum step height of 310mm) by pitching up the front of the robot to enable the front wheels to get grip on the obstacle (fig. 1-4a). In a similar way, gaps with a span of over 50% their body length (with a maximum span of 400mm) can be crossed by pitching up the front of the robot and in this way displacing the centre of gravity (fig. 1-4b). An actuated yaw motion is used to steer the robot between obstacles. The last version of the series, the SOURYU-IX, also incorporates an active rolling motion halfway down its body. This is used to create grip with the wheels when the robot is stuck in a narrow ditch. By activating the motor, the two body halves are twisted and the wheels are pushed against the walls. This creates traction which enables the robot to move out of the ditch.



(a) Souryu-VII climbing over a step of 180mm.

(b) Souryu-VIII climbing over a gap of 250mm .

Figure 1-4: Special rough terrain climbing abilities of the Souryu robots.

1-4-1 Centipede Zebro

The previously mentioned studies have shown that long flexible bodies with many legs perform well on rough terrain. An actuated flexible spine can also increase the ability of a robot to climb steps and gaps, which is an important part of navigating over rough terrain. The disadvantage of this body morphology is that these robots tend to get more complicated and expensive when a long body with many legs is used. Each leg often needs a separate motor and controller, increasing the costs. Also, it is not always necessary to have a long body. Only when confronted with certain types of obstacles like gaps, steps or highly uneven terrain, a long body increases rough terrain capabilities. On relatively flat terrain, a short body with only six legs is sufficient. This is why the idea came to mind to create a swarm of short six-legged robots capable of moving over medium-rough terrain by themselves, with the added ability to connect together and form a chain. This chain of robots will behave like a centipede with up to 30 legs, and be able to overcome obstacles that the individual six-legged robots cannot overcome. The idea of a modular centipede robot is based on the general concept of Modular Self-Reconfigurable (MSR) robots and swarm robots. There have been over two dozen studies on MSR robots and an overview of the most promising models is given by Yim et al. [20]. Some well-known MSR robotic platforms are the Modular Transformer (M-TRAN) [21] and the Connector Kinetic Robot (CKBot) [22]. Their modular nature allows the robots to change their morphology for a specific task. The M-TRAN can form several different morphologies, ranging from small and large quadrupeds to snake-like configurations and even a cartwheel configuration (fig. 1-5). The CKBot can also form a legged configuration with compliant legs attached to the modules which allows it to walk dynamically. This modular nature seems ideal, but it also has some drawbacks. Most modular robots consist of small building blocks with up to six DOFs, driven by standard hobby servos or small motors. The building blocks are shaped in such a way that the robot can form various morphologies. This is good for diversity, but it also limits the strength of the robot. The small modular blocks only have small motors and are not built for a particular purpose. This is why a purpose-built rough terrain robot will always outperform a modular robot. The golden mean would be a purpose-built rough terrain robot with the ability to also change its shape for specific

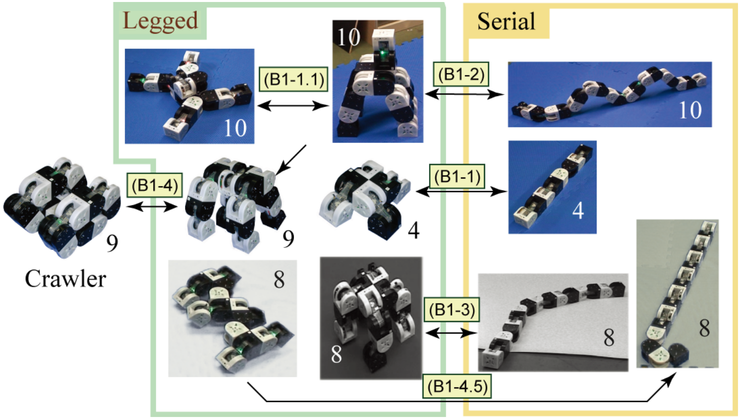


Figure 1-5: M-Tran III MSR robot in different configurations.

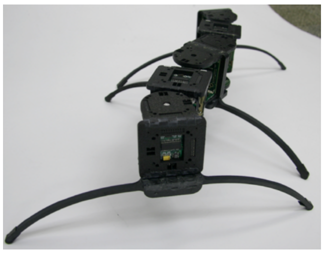


Figure 1-6: CKbot legged configuration.

obstacles. Such a robot was proposed in a paper by Miner et al. [23]. This paper describes a modular centipede robot with the ability to change its length by autonomously connecting multiple modules together. The study only describes the robot in a simple way as more of a general idea but with no technical details or tests. There are also at least a couple of centipede robots already in existence, but they have a fixed morphology [24] [25] [16].

1-5 Problem Statement

The idea of a swarm of independent USAR robots with the ability to connect together and form a long centipede seems promising, but also raises some of questions:

- The separate modules can be connected together with either a rigid connection or by incorporating joints to give the connection one or more DOFs (from now on the connection between two modules will be called the spine). How does adding DOFs in the spine influence the robots ability to move over rough terrain?
- The long centipede can be formed by a small or large number of modules. What is the influence of the number of modules on the rough terrain performance?
- Does the length of the spine influence the rough terrain performance?
- When joints are incorporated in the spine, they can be passive if they have a fixed stiffness, or active if they have an actuator that can be controlled. Is there an advantage in using active joints for certain obstacles?
- When using active joints in the spine, how should they be controlled for different types of obstacles?

To answer these questions, a simulation environment was used to simulate different centipede robot morphologies and test their ability to overcome obstacles found in disaster areas. The modular centipede system consists of separate legged units that are designed to move over rough terrain individually, but are also capable of connecting together with the use of a rigid or flexible spine system. The flexible spine system can either be active or passive and contains one or two DOFs. This modular centipede design idea led to the following hypothesis:

> "Introducing compliance in the spine and increasing the number of legs on a centipede robot improves the locomotion performance over rough terrain."

This hypothesis can be divided into two parts regarding the spine and the number of legs. The first sub-hypothesis focusses on the effect of the spine and says that:

> "Variable compliance in the spine improves mobility over rough terrain."

Here variable compliance must be seen as either different stiffness levels of the spine when the joints are passive, or different motor actuation strategies if the joints are active. The second sub-hypothesis regards the number of legged units in the modular centipede, and says that:

> "Increasing the number of legs will improve mobility over rough terrain."

This means that adding units to the centipede robot will enable it to overcome more difficult terrain and greater obstacles. The terrain types and obstacles used for testing will be discussed in section 2-2-4. The hypothesis and sub-hypotheses will be tested with extensive simulations as will be discussed in greater detail in chapter 2.

1-6 Scope and Outline

This thesis report consists of two main parts. The first part introduces all the preliminary knowledge used for this project. Chapter 1 gave an overview of the research that has already been conducted on USAR robots and introduced the idea of the Modular Centipede Zebro. The second part contains the added contributions and consists of 4 chapters. Here, all details about the extensive simulations, design choices and control strategies used to test the hypothesis can be found. Chapter 2 explains how the centipede robot is simulated with the use of the robotic physics simulator V-REP. In chapter 3, the actuation of the spine and legs for various obstacles will be discussed. Chapter 4 will reveal and discuss the simulation results and chapter 5 contains a general discussion about the concept of the Modular Centipede Zebro.

Part II

Added Contributions

Chapter 2

Simulation Environment

In order to establish a first design for the Modular Centipede Zebro (MCZ) robot, it can be advantageous to test different configurations in a simulated environment. In these simulations, different spine lengths can be tested, as well as motor torques, various active and passive degrees-of-freedom (DOFs) and different robot lengths. The main contributions of this thesis project are the extensive simulations conducted to test all these parameters on various obstacles. In this chapter, the set-up of these simulations will be explained. The results of the simulations can be found in chapter 4.

Simulation Software There are a couple of different simulation software environments available, like SimMechanics (from Mathworks), 20-sim, Webots (from Cyberbotics) and V-REP (from Coppelia Robotics). Most of them are commercial software packages which can be fairly expensive and except for SimMechanics they are not part of the standard software library of the Delft University of Technology (TU Delft). This is why the choice for simulation software was between SimMechanics, for which the TU Delft has a license, and V-REP, which is open source and free to use for educational purposes. Since V-REP seems to perform well with contact dynamics simulations, and it is also easy to learn and use with many tutorials available, it was chosen as the simulation environment for this thesis project.

2-1 Modelling in V-REP

2-1-1 About V-REP

V-REP is an elaborate open source free-ware robot simulator, made by a company called Coppelia Robotics. It is based on a distributed control architecture: each object/model can be individually controlled via an embedded script, a plug-in, a Robot Operating System (ROS) node, a remote Application Programming Interface (API) client, or a custom solution. It has an integrated development environment with which models can be made or adjusted. Models can also be imported in different file formats. Since V-REP uses triangular meshes to describe and display shapes, it will only import formats that describe objects as triangular meshes. It supports OBJ, DXF, 3DS and STL file formats. The latter is a popular format used by different Computer-Aided Design (CAD) programs. V-REP's dynamics module currently supports three different physics engines: the Bullet physics library [26], the Open Dynamics Engine [27], and the Vortex Dynamics engine [28]. The first two are open source and free to use, for the Vortex engine a license is required. For this thesis project only the Bullet engine will be used.

Models can be controlled in different ways with V-REP. A script can be attached to a model and with the use of regular API functions, joints can be controlled, parameters can be read and almost everything can be adjusted. The language used for programming scripts in V-REP is Lua [29]. This is a powerful, fast, lightweight, embeddable scripting language which was developed at the Pontifical Catholic University of Rio de Janeiro in Brazil. It is also possible to control robot models trough a remote API. With a remote API, 100 of the total 400 functions can be called by either a C/C++ application, a Python script, a Java application, a Matlab program, or an Urbi script. To keep the simulations robust and simple, and because not all the functions are available in the remote API library, only scripts from within V-REP will be used for this thesis project. This means that all scripting will be done in Lua.

2-1-2 V-REP Models

In order to run dynamic simulations in V-REP, first the models need to be created. This can be done directly in V-REP, or by importing CAD models. Imported models need minor adjustments in order to work in V-REP. Also, simulated actuators need to be added. How this process works is described in detail in appendix B.

2-2 Models used for Simulations

To test the influence of certain body morphologies on the ability of the MCZ robot to navigate over various obstacles, several models were built in V-REP. These models are based on the already existing Zebro robot, which is used for research at the TU Delft. How these models were designed in V-REP will be explained in the following sections.

2-2-1 Zebro Model

The Zebro robot, which is available at the TU Delft, is inspired by a robot called RHex, on which more information can be found in section 1-3. It possesses six actuated DOFs (at the hip of each leg), and it uses compliant C-shaped legs. The RHex design emphasizes on mechanical simplicity, which is important for rough terrain robots. Fewer moving parts and a robust design are necessary to prevent the robot from getting stuck or damaged. The design is symmetrical, so the robot is able to walk upside-down. With the compliant legs, the robot is able to run dynamically and there is evidence that it closely resembles the well-known spring-loaded inverted pendulum (SLIP) model [30] for dynamic walkers.

To build the Zebro model in V-REP, the original CAD model (made in Solidworks) was stripped down by removing parts which are not necessary for simulation, like PCB's, batteries, sensors, etc. This resulted in a stripped-down version of Zebro with only the basic shape of the body and the six legs. This basic Zebro model was exported to a STL-file and then imported in V-REP, resulting in one complete non-pure shape. By means of the triangle edit mode, the legs were separated to six non-pure shapes. Hereafter, the dynamic model had to be built by using pure primitive shapes. This is easy for the body, which can be modelled by a simple cuboid, but the legs are C-shaped which makes it impossible to create a model out of only one primitive shape. To match the shape of the legs, several cuboids were connected, resembling the legs in shape and thickness. The original Zebros legs are compliant. It is not yet possible to easily simulate compliant components such as these legs in V-REP, so the model used for simulation uses rigid elements in the legs. This means that the model will not be exact in terms of energy storage in the legs, but since the robot will be walking over rough terrain and power consumption is not measured, this simplification should not make a large difference in terms of performance. The non-pure shaped model and the dynamic pure shaped model can be seen in fig 2-1. The dimensions of the body are 440x240x67,5 mm and when the robot is standing on its legs, its total height is 184 mm. The weight of the model is set to 7 kg, which is distributed evenly as if the robot was made from a homogeneous material. These measurements closely resemble the real Zebro. In V-REP, different material models can be used, each with their own properties. For the Zebro and MCZ models, two different material models were used of which the properties are listed in table 2-1 and 2-2. The bodies, spine and all obstacles are made from the default material (table 2-2), and the legs are made from the high friction material (table 2-1) since the real Zebros legs are also made from high friction materials like rubber.

Material name: highFrictionMaterial	
Friction	1.00
Restitution	0.00
Linear damping	0.00
Angular damping	0.00

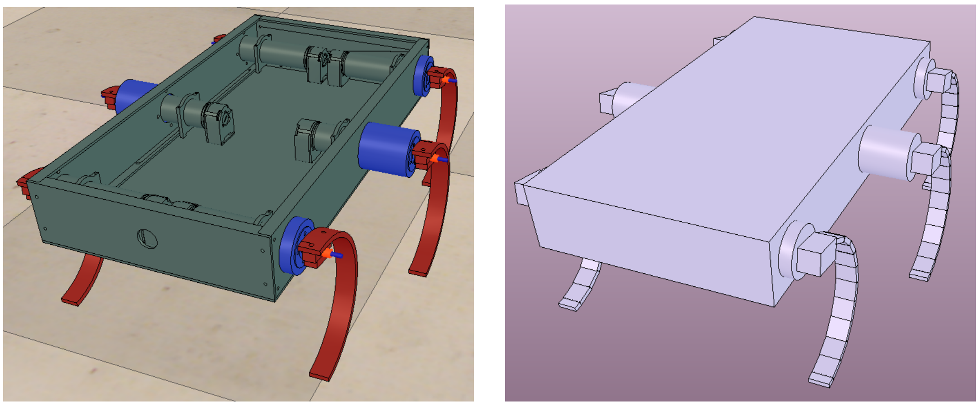
Table 2-1: V-REP high friction material properties.

Material name: defaultMaterial	
Friction	0.71
Restitution	0.00
Linear damping	0.00
Angular damping	0.00

Table 2-2: V-REP default material properties.

2-2-2 Rigid Centipede Zebro Model

The MCZ models are a way to test different configurations based on the original Zebro. They are formed by connecting separate modules with only 2 legs with the use of a spine. In this way, the amount of legs on the robot can be varied easily. One module has a dimension of 120 x 240 x 67,5 mm and uses the same legs as the normal Zebro model, giving it a total height of 184 mm when standing. The weight of a module with 2 legs is set to 2kg, which should be a realistic weight for such a device. For the Rigid Centipede Zebro (RCZ), the building blocks are connected by a rigid weightless cuboid. The connection is chosen to be weightless, since there are no mechanical details available for this connection. This thesis project only focuses on the morphology of the robot and not so much on the mechanical design. A V-REP add-on script was written to easily create different versions of the RCZ. With this add-on, the number of legs can be changed by simply clicking one button. The script places the modules in the right position, creates the rigid spine, and connects everything together, creating a



(a) Non-pure shaped Zebro model.

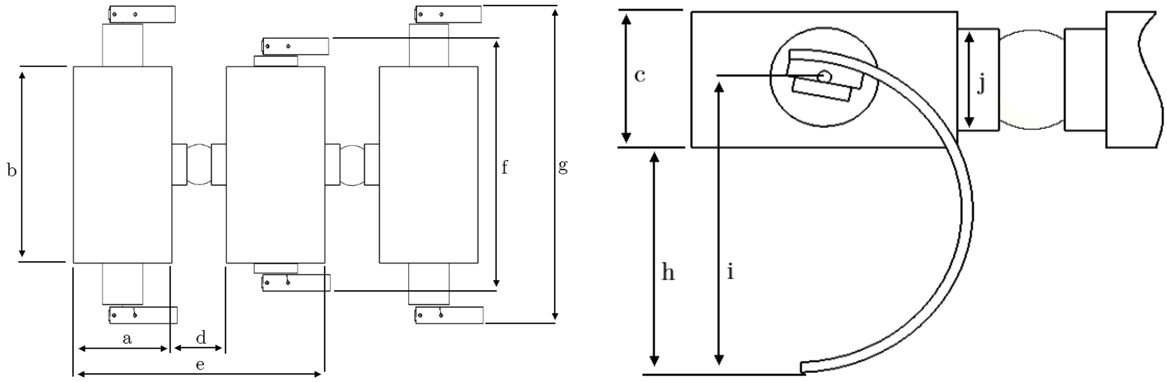
(b) Pure-shaped dynamic Zebro model.

Figure 2-1: Overview of the Zebro model used in simulations.

complete working model. The length of the spine is user-defined and is initially set to match the dimension of the original Zebro. However, it is later changed in an experiment to test the influence of the spine length. Both the RCZ and the Flexible Centipede Zebro (FCZ) use exactly the same dimensions, except for one longer version of the RCZ which uses a spine of 134 mm instead of 67 mm. The dimensions of the MCZ models can be found in figure 2-2.

2-2-3 Flexible Centipede Zebro Model

The FCZ model is built in the same way as the RCZ model, but instead of a rigid connection between the modules, one or two DOFs are incorporated in the spine. This is realized by cutting the rigid connection in half and placing joints in between. There are different versions of the FCZ, which will permit different types of rotation. The joints can be built in such a way that they will permit only roll, pitch, yaw, or a combination of those motions. The minimum angle and motion range of the joints can be defined, as well as the stiffness and damping coefficients. For the simulations in this thesis project, only pitch and roll motions are enabled. The literature study preceding this project has shown that pitch and roll motions can increase the performance on rough terrain, while yaw motions are only used for steering. Each version of the FCZ robot has both pitch and roll joints built in, but in some cases, one of the joints is locked by adjusting the motion range to 0. This effectively eliminates the motion in this direction. Five different versions of the passive FCZ are tested. Pitch and roll motions are simulated separately, as well as in a 2 degree-of-freedom (DOF) combination with both high and low stiffness. A special case of a 1 DOF pitch motion is also tried, where the joints can only rotate in a limited range. The joint parameters for the different versions of the FCZ can be found in table 2-3. In addition to the passive FCZ models, also active spine joints are simulated. More on these can be found in chapter 3. In figure 2-3, the FCZ model with different lengths is placed next to the original Zebro model for comparison.



(a) Centipede Zebro top view.

(b) Centipede Zebro side view.

Dimension	a	b	c	d	e	f	g	h	i	j
Size [mm]	120	240	67.5	67	307	308	387	110	147	50

Figure 2-2: Dimensions of the MCZ model.

	FCZ version	1	2	3	4	5
Pitch Joint	Minimum Joint Angle [°]	-20	0	0	-20	-20
	Motion Range [°]	40	0	20	40	40
	Spring Constant [Nm/rad]	1.000	-	1.000	10.000	1.000
	Damping Coefficient [N · s/rad]	15	-	15	15	15
	Maximum Torque [Nm]	4.000	10.000	4.000	4.000	4.000
Roll Joint	Minimum Joint Angle [°]	0	-20	0	-20	-20
	Motion Range [°]	0	40	0	40	40
	Spring Constant [Nm/rad]	-	1.000	-	10.000	1.000
	Damping Coefficient [N · s/rad]	-	15	-	15	15
	Maximum Torque [Nm]	10.000	4.000	10.000	4.000	4.000

Table 2-3: Joint parameters of the different FCZ versions. 1 = 1-DOF pitch, 2 = 1-DOF roll, 3 = 1-DOF limited pitch, 4 = 2-DOF high stiffness, 5 = 2-DOF low stiffness

2-2-4 The obstacles

In a disaster area, Urban Search and Rescue (USAR) robots may encounter various obstacles. These include uneven terrain caused by debris from demolished buildings, steps, gaps, fire, water, etc. For this thesis project, only the first 3 obstacles are simulated. For each type of obstacle, objects are built in a simulated scene, as will be explained in the following paragraphs.

Fractal Landscape A large part of a disaster area consists of rubble from destroyed buildings. This creates an uneven landscape with alternating high and low points. In research, this is often simulated by placing wooden blocks of different lengths next to each other to create a rough landscape. This technique is also used in the Robocup Rescue competition. For the simulations, a similar approach is used. A fractal landscape is created by placing cuboid

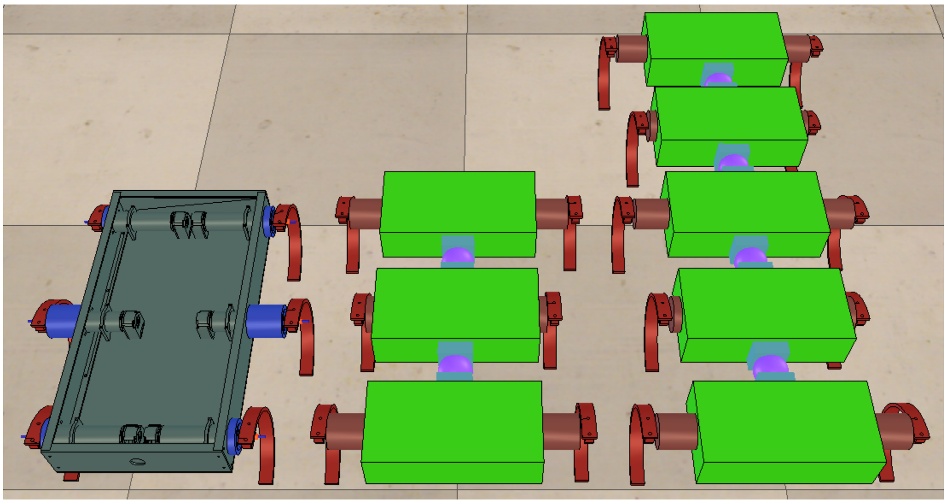
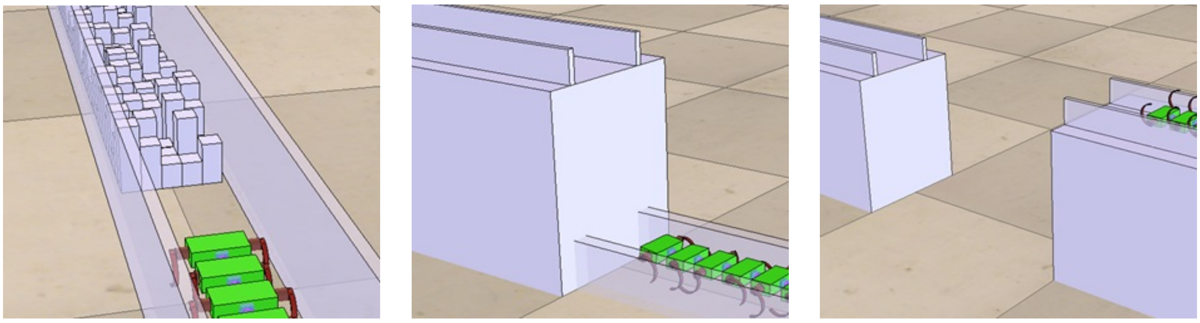


Figure 2-3: Line-up of the original Zebro (on the left) together with a 6-legged and a 10-legged version of the FCZ.

shapes with dimensions of 100x100 mm and different heights next to each other. The height is different for each block and is generated with the random function of the Lua programming language. This gives the height of the blocks an even distribution. The random function generates a random number between 0 and 1 which is afterwards multiplied by a constant value between 0 and 0.60. This constant multiplication factor will define the difficulty of the terrain. When it is set to 0, the terrain will be completely flat, and when it is set to a certain value, this will be the maximum height (in meters) of the blocks. Since the average of the random generator is 0.5, the average height of the terrain will be half of this multiplication factor. The terrain will be 5 blocks (or 50 cm) in width and 250 blocks (or 25 meters) in length. Two high walls are placed on the sides, so the robot is contained within the test terrain. By using the walls to direct the robot forward, no other steering control is needed. In figure 2-4a the simulated terrain is visualized.

Step Another obstacle often found in a disaster area is a large step. The height of the step can range from just a few centimeters to several meters. A step is simulated by placing a cuboid with a certain height between two walls and placing the robot in front of it. The robot has to reach the elevated platform to succeed. The height of the cuboid is a measurement for the difficulty of the obstacle (fig. 2-4b).

Gap When two platforms have a gap in between, this can pose a difficult obstacle for a robot to overcome. In a case of failure, the robot will fall in between the platforms and won't be able to get out, rendering it useless. This obstacle can be simulated by generating two cuboids with a certain distance in between them. The robot is placed on the first platform and is contained by walls on the sides. The objective is to get to the second platform without falling in the gap. The distance between the platforms is a measurement for the difficulty of the obstacle (fig. 2-4c)



(a) Simulation of uneven terrain.

(b) Simulation of a step.

(c) Simulation of a gap.

Figure 2-4: Simulation of different obstacles.

2-2-5 Simulation Process

A large part of the simulation process is automated with the use of an add-on script. This script, which is written in Lua code, runs in the background and overviews the simulation process. While the gap and step obstacle simulations are started and tuned manually, the fractal landscape simulations are controlled by this script. For the fractal landscape simulations, first a scene has to be loaded manually. The scenes are created by placing a MCZ robot in a standard scene with a 5 x 50 m long flat floor. When the scene is loaded, the script will place the MCZ robot in the starting position. The length of the robot is taken into account in such a way that independent of its length, each robot will start with the front of its first module in the same position. After the robot is placed in its starting position, the fractal landscape is created. A field of 5 by 250 blocks of 100 x 100 mm is placed on the floor, where the height of each block is randomized with an even distribution between 0 and 1, which is then multiplied by a constant to set the difficulty. This constant ranges from 0 to 60, resulting in an average height of the fractal landscape ranging from 0 to 0.30 m (in this last case, the maximum block height will be 0.60 m). A simulation is then started, and the robot is dropped in the fractal landscape. With the first difficulty level set to 0, this results in a flat ground for the robot to walk on. The simulations run for 60s with a 50ms time step (20 simulation steps per second). At each time step, the x, y, and z positions of the robot are stored. When the 60 s time limit is reached, the simulation will stop and the final distance that the robot has travelled is stored in a file, together with information about the robot and the simulation (body type, number of legs, time step, final time, difficulty level, etc). Hereafter, the robot is placed in its starting position again, and a new random landscape with the same difficulty level is created. This process is repeated 200 times. After these 200 simulations, the difficulty level is increased by 5cm, increasing the average block height with 2.5cm. Then another set of 200 simulations is started. This process is repeated until the final difficulty level of 60 is reached. This script is used for all the different robot versions, each with a number of modules ranging from 3 to 7 (6 to 14 legs). There are 8 different MCZ robot versions (2 rigid, 5 passive compliant and 1 active compliant), each with 5 different lengths. With the terrain difficulty ranging from 0 to 60 with increments of 5 (13 difficulty levels in total), and 200 simulations per difficulty level, this means that in total 104.000 simulations were conducted for the fractal landscape obstacle. Each simulation takes about 40 s from start to finish, which means that the total simulation time would come to 1156 hours. Luckily, it is possible to load multiple instances of V-REP on a multi-core computer and share the load on the cores. With the use of two quad-core computers running 24/7, all simulations were finished within 2 weeks. This resulted in 8 GB of data files, which were processed by a MATLAB script to get an overview of the results. More on this can be found in chapter 4.

For the step and gap obstacles, the simulations were conducted manually. First, the MCZ robot is placed in a scene with the particular obstacle. Starting with the longest version of the robot, one with 15 modules (30 legs), the difficulty of the obstacle is increased until the maximum difficulty that the robot is able to overcome is found. The spine of the robot is actuated in different ways to increase its ability to cross these obstacles, on which more information can be found in chapter 3. The parameters of this spine actuation script are manually tuned, until it can be said with confidence that the robot has reached its absolute maximum difficulty level. This difficulty level is noted, and next the length of the robot is decreased by 1 module. With the new shorter robot, the same process is repeated and the difficulty level is lowered until the robot is able to conquer the obstacle. The length of the robot is decreased until the minimum length of 3 modules is reached (6 legs). The results can be found in chapter 4.

Chapter 3

Actuation

The Modular Centipede Zebro (MCZ) robots performance is tested in simulation against different obstacles. To overcome these obstacles, the legs need to be actuated and controlled. Some of the obstacles also require the spine to be actively controlled. To acquire smooth movement of the spine and legs, two separate controllers are implemented. How these controllers work and which control sequences are used will be explained in this chapter.

3-1 Leg control with gait switching

A multi-legged robot has a certain order in which the legs move. This is called the gait. If for example a hexapod robot needs to move forward, but also remain stable at each point during the movement, it has to keep at least three legs on the ground at all times. An example of such a tripod gait is given in figure 3-1. During the first time period the robot lifts legs 1, 4, and 5, meaning they are in swing phase. This lasts for a time period τ_f called the flight time. During this period, the stance legs (2, 3, and 6) remain in contact with the ground while rotating, pushing the robot forward. When the first set of legs touches the ground again, all legs remain in contact with the ground for a time period called the double stance time τ_Δ (while still pushing the robot forward). Then the second set of legs lifts off while the first set remains stance for a time period τ_g, called the ground time. The gait $\mathcal{G}$ in which the robot moves can be described by groups of integers representing leg pairs that move together. The previously described tripod gait can be mathematically represented as $\mathcal{G}_{tripod} = \{1, 4, 5\} \prec \{2, 3, 6\}$. Another example of a gait for a hexapod robot is the quadruped gait as seen in figure 3-2, which is described as $\mathcal{G}_{quadruped} = \{1, 4\} \prec \{3, 6\} \prec \{2, 5\}$. During movement, the robot is able to switch between different gaits and adjust the timing of the lift-off and touch-down moments of its legs on the fly. This is achieved with a method called max-plus algebra [31] [32]. The gait $\mathcal{G}$, ground time τ_g, flight time τ_f, and double stance time τ_Δ are used as inputs, and a schedule with the lift-off and touch-down times is returned. By using a **continuous time scheduler**, the discrete events calculated with max-plus algebra are converted into a reference signal for the leg controllers by interpolating between touch-down and lift-off angles (θ_t and θ_l), returning the desired leg angles at each moment in time.

A detailed description of how max-plus algebra and the continuous time scheduler work can be found in appendix C. To keep the focus on the body morphology and eliminate the effect of the leg actuator dynamics, the legs are assumed to always be powerful enough to reach the desired angle. Therefore, the leg joint controllers are tuned in V-Rep in such a way that they always keep their trajectory and are strong enough to propel the robot in every situation. For a real-life robot, the leg actuators form a major part of the robots performance and can certainly not be neglected. Therefore, further research is necessary to test the effect of the leg actuators on the rough terrain performance of the robot.

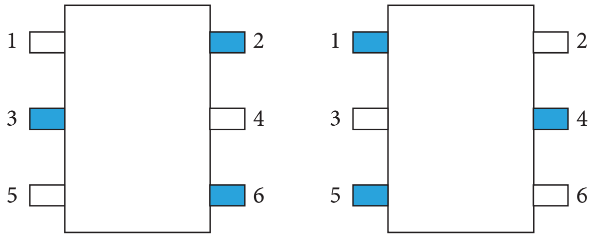


Figure 3-1: Example of a tripod gait on a hexapod robot.

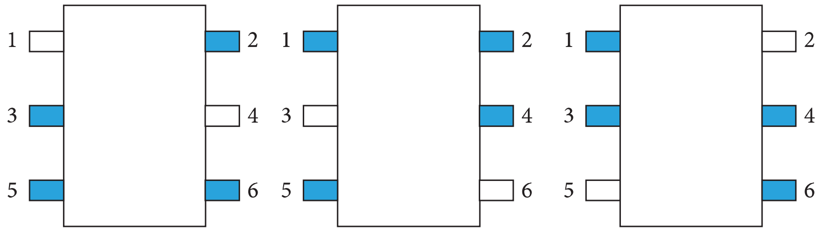


Figure 3-2: Example of a quadruped gait on a hexapod robot.

3-2 Spine Control for Gap and Step Obstacles

For the gap and step obstacles, the MCZ robot is required to actively move its spine. In V-REP, the spine is modelled as two rigid blocks with two joints connecting them in order to enable movement in the pitch and roll directions. Since the gap and step obstacles only require the robot to move in a 2D-plane, only the pitch joints are controlled while the roll joints are locked. Joints can be controlled in V-REP by using a script to send the target positions to the joints, after which a PID-controller in each joint applies the necessary forces to achieve the target position. These joints can be further configured with a maximum torque, a motion range, a maximum velocity, and the P, I, and D constants for the controller. For the different obstacles, these parameters vary and will be supplied in the form of a table in this chapter.

In order to climb a step or cross a gap, the robot must bend its spine to create a shape which enables the legs to reach either the top of the step or the second platform across the gap. This shape is different for each of the two obstacles, and also varies for different obstacle heights and widths. It is important that the robot adapts to the required shape smoothly, in order to minimize disturbances from the accelerations in the spine joints. This is done by using two different functions. The first function creates the required shape by assigning an integer to each module of the robot, which corresponds with the position it must move to. A second function then reads out the required shape and gradually moves the spine joints to their target positions by sending the joint angles to the individual joint controllers. To clarify this process, an example is given in the following paragraph.

Suppose that the desired shape the MCZ robot has to form is as shown in figure 3-3. For the robot to be able to move smoothly into this position, a step by step approach is used. The initial condition shows the robot in a straight position (all spine joint angles set to 0°)(fig. 3-4a). The first control function describes the shape of the robot with the use of a column vector. Each module of the robot corresponds to a column in the vector. In that way, the straight position of the robot would be presented by the vector

$$v_{shape} = [1, 1, 1, 1, 1]$$

since all the units are in their initial position (fig. 3-4a). At the next step, the first spine joint takes on the angle of the last spine joint in its final shape. This means that the vector representing the shape gets a 2 on column 1, and every other column shifts one place, making the shape-vector

$$v_{shape} = [2, 1, 1, 1, 1]$$

A second control function reads out the shape-vector and sets the joint angles. It will move the target position of the spine joint between the first and second unit incrementally to 25° during a predefined time period (fig. 3-4b). This ensures the movement is smooth and with little jerk. In the next step, a 3 is added to the first column of the shape vector, making it

$$v_{shape} = [3, 2, 1, 1, 1]$$

The control function then moves the first joint from 25° to 60° and the second joint from 0° to 25° (fig. 3-4c). This process continues until the robot reaches its final shape (fig. 3-4d-f).

$$v_{shape} = [6, 5, 4, 3, 2]$$

Using this method ensures that the robot moves smoothly over the obstacles, as will be demonstrated later in this chapter.

Joint parameters The simulated joints in V-REP can be configured with a couple of parameters. The motion range can be set with a position minimum (this is the lowest angle the joint can take), and a position range, which essentially sets the position maximum. There is also a maximum step size (the maximum difference in joint angles between two simulation steps), but this will remain constant at the default of 10 degrees per step for all simulations. The dynamic parameters consist of the maximum torque that the joint can apply, the maximum velocity at which the joint can rotate, and the P, I and D gains for the joint controller. For the simulations where the spine is actuated, two parameter sets are used. To keep the simulations realistic, the force that the joints can generate is limited. If the MCZ robot would actually be built, the weight and power of the spine actuators would be important factors to keep in mind during the design process. This limits the motor size and maximum torque that can be used. For the simulations, two parameters sets are used, representing possible

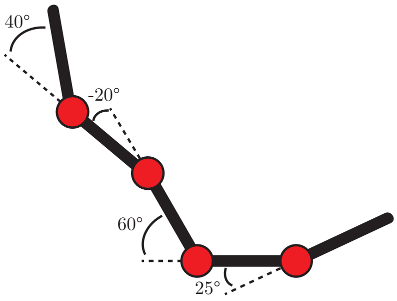


Figure 3-3: Example shape for a MCZ robot with 5 modules.

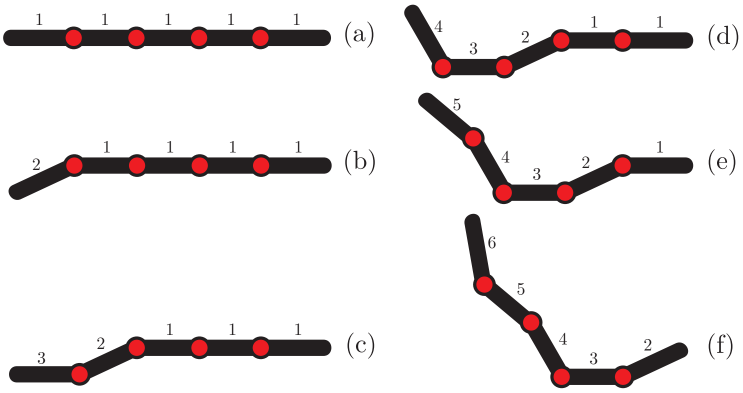


Figure 3-4: Step by step process in which the MCZ robot moves into a certain shape.

	Joint Parameter Sets	
	Set 1	Set 2
Position Minimum [°]	-50	-50
Position Range [°]	100	100
Max. Torque [Nm]	40	5
Max. Velocity [°/s]	120	120
P-Gain	80	80
I-Gain	0	0
D-Gain	0,5	0,5

Table 3-1: Joint parameter sets.

real life scenarios. The first parameter set uses a maximum joint torque of 40 Nm, and the second set uses a maximum torque of 5 Nm. A joint torque of 40 Nm is quite high for a small robot, but it is achievable with the use of a stepper motor in combination with a planetary gear box. Other options would be to use a worm-wheel to reduce backdrivability, or to use hydraulics (although this would not be very practical for this type of robot). The second parameter set uses only a 5 Nm maximum torque, which is easier to achieve, but this also limits the capabilities of the robot as can later be seen in the results. All the parameters of the two configurations can be found in table 3-1. The P, I, and D gains were tuned manually in V-Rep until the movement of the spine joints was smooth and reliable. During the gap and step obstacle simulations, the roll joints are locked by setting the minimal position to 0, the motion range to 0, and disabling the controller.

3-2-1 Gap Obstacle Spine Control

First gap crossing method For a robot to be able to cross a gap, three moves are essential. First the robot has to place its front legs on the second platform. When this succeeds, the robot must move as many legs as possible over to the second platform before the last pair of legs leaves the first platform. Then it has to pull up the remaining legs, which dangle over the edge of the second platform. For the simulations, two different methods are used to get the front legs on the second platform. For the first method, the spine joints between the first five modules are gradually set to 15°, which raises the front legs while the robot walks forward and displaces its center of gravity backwards (fig 3-5a). A synchronous gait with two leg sets is used (opposite legs moving simultaneously), to keep oscillations to minimum. An example of this gait would be

$$\mathcal{G} = \{\{1,2,5,6\},\{3,4,7,8\}\}$$

While moving forward, the first two legs remain stationary in a forward position, to increase the distance that the robot can reach. When the front legs are above the second platform, all spine joints are set to 0° so the front of the robot falls down, landing its front legs on the platform. From here on, the first two legs join the synchronous gait. When the last module leaves the first platform, the robot continues to walk forward, pulling up the dangling parts of its body. During this last move, the gait is changed to an asynchronous alternating gait (opposite legs move after each other). Simulations have shown that this gait change reduces slipping of the legs and enables the robot to pull up more modules. An example of this gait would be

$$\mathcal{G} = \{\{1,4,5,8\},\{2,3,6,7\}\}$$

The advantage of this first gap crossing method is that it is fairly simple. The disadvantage is that this method generates high torques in the spine joints. Since each module weighs about 2 kg with a centre of mass located at a distance of 0.094 m from the joint, lifting the first unit requires a minimal torque of $2 * 9.81 * 0.094 = 1.84$ Nm. The static load on the second joint when it is about to turn (legs of the second unit in flight phase) comes to $(0.094 * \cos(15) + 0.187) * 2 * 9.81 + 2 * 9.81 * 0.094 = 7.29$ Nm. Using basic geometry, it can be calculated that the static loads are 16.10 Nm, 27.80 Nm, 41.71 Nm for the third, fourth and fifth spine joints. Adding to this the fact that there are also dynamic loads because of bouncing of the robot due to the movement of the legs, the load quickly becomes too large for the motors. The simulations show that the robot already has difficulties with turning the fourth joint when parameter set 1 is used (40 Nm max. torque), and it will not even turn the second joint when parameter set 2 is used (5 Nm max. torque). In figure 3-6 the simulation of the first method can be seen. The results will be discussed in section 4-1.

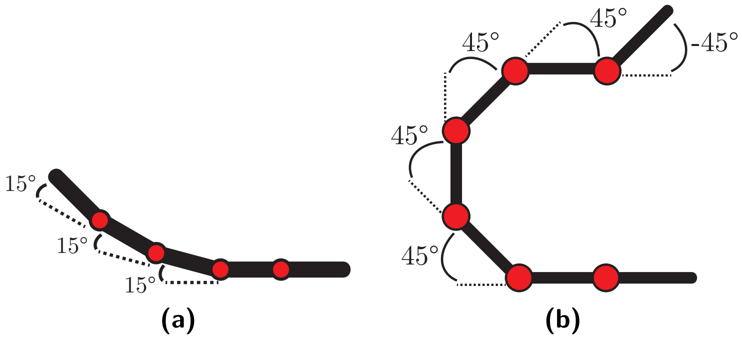


Figure 3-5: Spine shapes used for the first (left) and second (right) gap crossing methods.

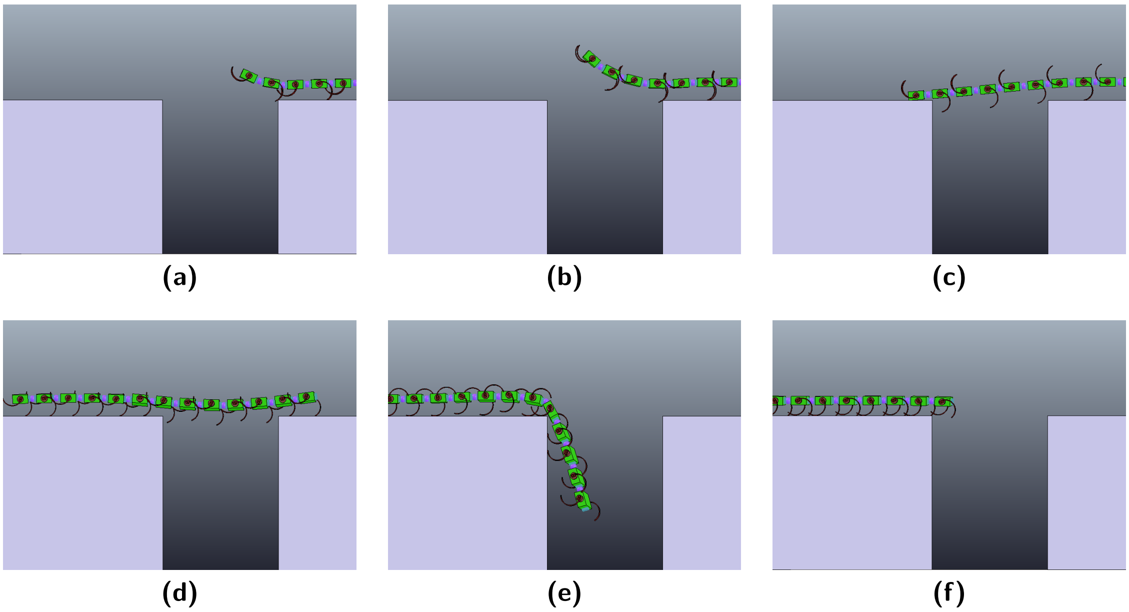


Figure 3-6: Overview of the first method to cross a gap.

Second gap crossing method Because of the large torques required for the first method, a second method with lower torques was tried. With this method, the joints are turned to 45° so an overhang is created with the first module hanging over the sixth spine joint (figure 3-5b). The robot smoothly moves into this shape with the previously mentioned control functions. While remaining in this shape, the robot walks forward in a synchronous gait and keeps its front legs forward to maximize its reach. When the projected distance it can reach is greater than the distance between the platforms, all joint target angles are set to 0° and the robot shoots forward so the front legs reach the second platform. The robot then continues to walk forward. After the last unit leaves the first platform, the robot pulls itself up in the same way as with the first method (using an asynchronous gait). By using basic geometry to calculate the static loads on the joints, the following torques are found: 1.84 Nm on the first joint, 6.81 Nm on the second, 13.07 Nm on the third, 18.04 Nm on the fourth, 19.88 Nm on the fifth and 18.58 Nm on the sixth. These torques are lower than with the first method, and simulations show that the robot is able to complete the whole shape with parameter set 1 (40 Nm max. torque). When it stretches its body, 5 modules are shooting forward, enabling the robot to reach further than with the first method. The disadvantage is that this method is a little more complicated, since the distance that the robot is able to reach has to be predicted. When the robot shoots forward but comes up short, it will fall in the gap, while with the first method the front legs are already above the second platform when the legs are lowered. Another disadvantage is that the front module slams down at high speed on the second platform, which could cause problems. How this movement works in simulation can be seen in figure 3-7. The results of these simulations are discussed in section 4-1.

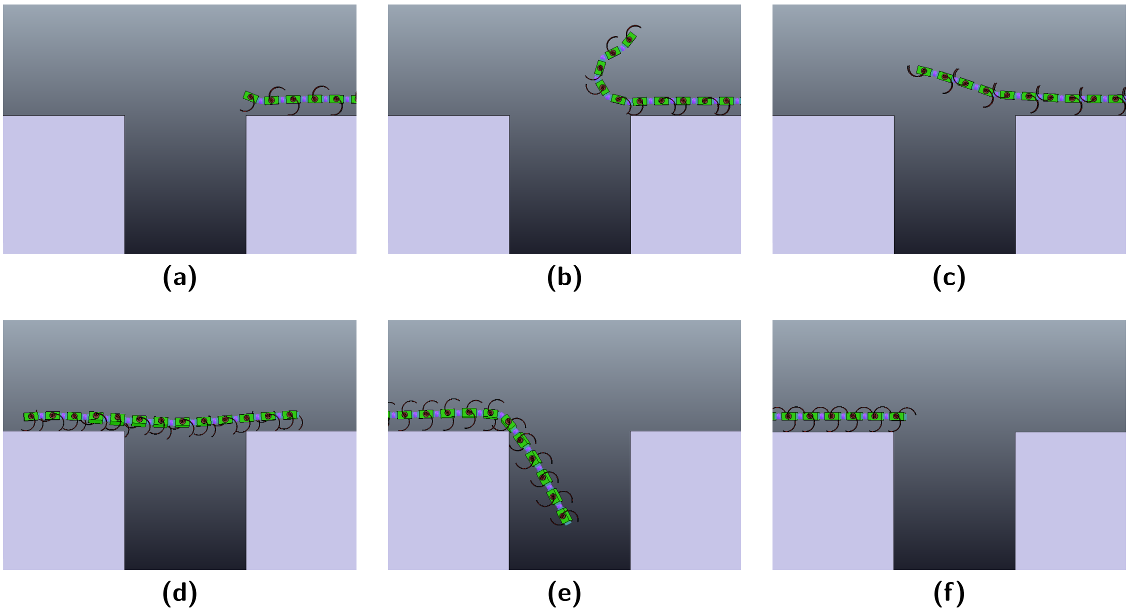


Figure 3-7: Overview of method 2 to cross a gap.

3-2-2 Step Obstacle Spine Control

In a similar way as with the gap obstacle, a step can be conquered by actively controlling the pitch joints of the spine. To climb a step, the robot has to get as many modules as possible up on the obstacle and then pull itself up. This is done by using the spine joints to push up the modules against the walls while the legs are used to keep the robot stable. The shape that the robot follows can be seen in figure 3-8, and is enforced by the same two control functions as for the gap obstacle. The amount of vertical modules (2 in this example) can be adjusted to match the height of the obstacle. During the climbing motion, the legs move in a synchronous gait in 3 sets. An example of this gait would be

$$\mathcal{G} = \{\{1,2,7,8\},\{3,4,9,10\},\{5,6,11,12\}\}$$

A synchronous gait is used to keep the robot stable while climbing. Simulations have shown that when the robot is almost at the top of the step, it easily falls over to the side because it is top-heavy. Using an alternating gait during the climb makes the robot unstable, decreasing the maximum height it can reach. Because of this reason, a synchronous gait is used during the climb. When the last unit leaves the ground, the gait is changed to an alternating gait in 2 sets. An example of this gait would be

$$\mathcal{G} = \{\{1,4,5,8,9,12\},\{2,3,6,7,10,11\}\}$$

Simulations have shown that using an alternating gait generates more pulling force, increasing the maximum height that the robot can climb. Figure 3-9 shows how the robot climbs an obstacle. Results from these simulations can be found in section 4-2.

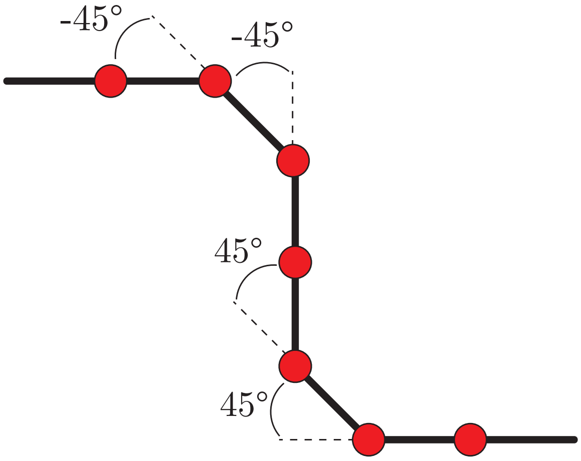


Figure 3-8: Spine shape used for climbing steps.

3-3 Fractal Landscape Spine Control

In addition to the 2 rigid and 5 passive compliant MCZ versions, also an active compliant version is tried on the fractal landscape. The idea is that continuously moving the pitch joints of the spine prevents the robot from getting stuck. If the robot gets stuck in the terrain and the movement of the legs is not enough to free it, the movement of the spine should enable it to break free and continue its course. The control scheme used for this movement is simple. The joints move continuously back and forth between $-30°$ and $30°$ with the use of a cosine function. The direction of the movement is alternated between the spine joints. This means that every uneven numbered joint moves according to

$$\theta_{joint} = \frac{30\cos(t * \pi)}{180 * \pi}$$

and every even numbered joint moves according to

$$\theta_{joint} = -\frac{30\cos(t * \pi)}{180 * \pi}$$

The results from these simulation can be found in section 4-3.

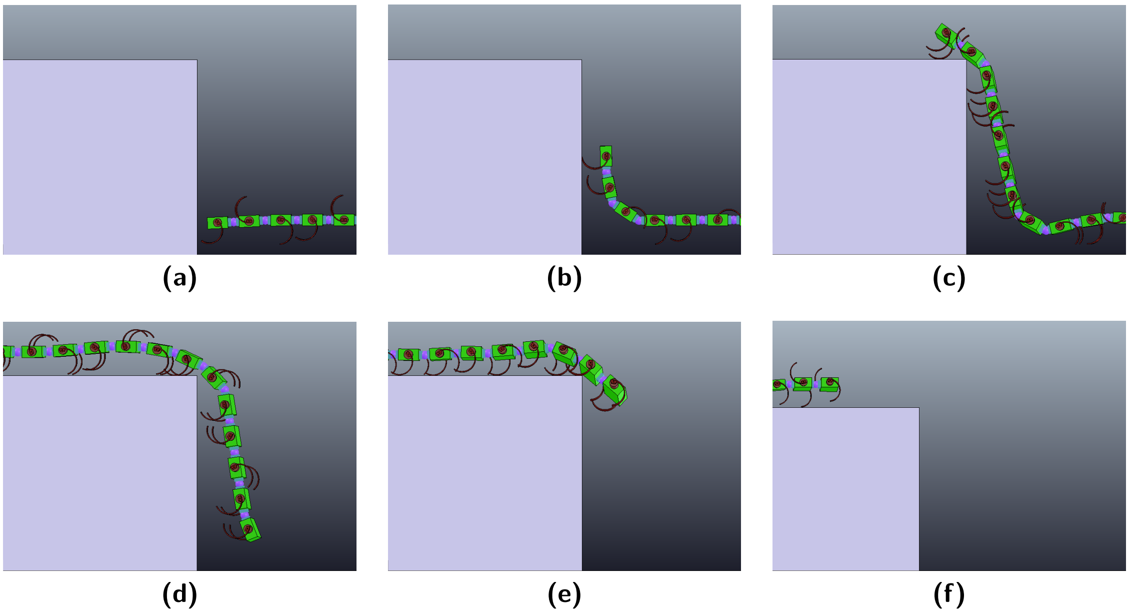


Figure 3-9: Overview of a the MCZ robot climbing a step.

Chapter 4

Results

A lot of information has been gathered during the numerous simulations that were conducted for this thesis project. The results from these simulations will be discussed in this chapter.

Data Processing All the simulation results from the different fractal landscape experiments are stored in over 500 separate m-files (simple text files which can be loaded in MATLAB). These files contain all the information about the simulations, like which Modular Centipede Zebro (MCZ) version is used, the number of legs on the robot, the current and final terrain difficulty levels, and the final position of the robot when the time limit of 60 seconds is reached. A MATLAB script is used to combine and process all these results, so that the different MCZ versions can be compared. First, the script browses through the folders containing the result files, and loads them into a cell. This cell contains 8 200x13x5 matrices for the 8 different MCZ versions. These matrices contain all the simulation results for each robot version (200 results per difficulty level, 13 difficulty levels, and 5 different robot lengths). During some simulations, the robot flipped over and walked in the wrong direction, resulting in a negative final distance. In a real-life situation, the robot will sense that it is upside down by using an Inertial Measurement Unit (IMU), and it will reverse the rotation direction of the legs to compensate for this. In the simulations, this mechanism is not implemented. To correct for these negative final distances, they are replaced with a distance of 0 m. The distance of 0 m counts as a penalty, since the robot flipped over. After the negative data points are replaced, the data is further processed. For each set of 200 simulations, the mean and standard deviation of the final distances are calculated and stored in a new cell. This cell contains 8 10x13 matrices for the 8 MCZ versions. On the uneven numbered rows, these matrices contain the means of the final distances for the 5 different robot lengths on each of the 13 terrain difficulty levels. The standard deviations are placed on the even numbered rows. These results are then exported to excel files where graphs and tables are created, which are used later in this chapter.

The gap and step obstacle simulations are started manually. Also, the controller parameters are hand-tuned. Since the initial conditions for each simulation are the same, it is not useful to run multiple simulations with the same parameter set. The parameters are tuned until it can be said with confidence that the maximum difficulty level for each robot length is found. The results are then written in a table. No further data processing is required for these experiments.

4-1 Gap Obstacle Results

For the gap obstacle, a MCZ robot with an actively controlled spine is simulated. Two different gap crossing techniques were used, on which more information can be found in section 3-2-1. The spine joints are limited to a rotation in the pitch direction, and 2 different parameters sets are used. The first parameter set uses a maximum joint torque of 40 Nm, while the second set limits the torque to 5 Nm. For the first method, the robot lifts up its front legs by setting the target positions for the first few spine joints to 15°. Since this generates a large torque in the spine joints, a second method is tried where an overhang of the first module over the sixth spine joint is realised to reduce the torques. In addition to the MCZ robot with an actively controlled spine, also 2 robots with a rigid spine are tested. The first rigid robot has a standard length spine (the legs are separated at the same distance as the standard Zebro robot at 187mm), and the second version has double the spine length (creating a leg distance of 254mm). More about the dimensions of the MCZ model can be found in section 2-2-2 and figure 2-2.

Active Pitch Control Centipede Zebro Mode 1

Number of Legs	6	8	10	12	14	16	18	20	22	24	26	28	30	Joint Torque
Body Length [m]	0,49	0,68	0,87	1,06	1,24	1,43	1,62	1,80	1,99	2,18	2,36	2,55	2,74	
Maximum Gap Width [m]	0,10	0,25	0,40	0,45	0,60	0,70	0,70	0,80	0,75	0,80	0,90	0,90	0,80	40 Nm
Percentage of Body Length [%]	20,2	36,7	46,1	42,7	48,3	49,0	43,3	44,4	37,7	36,7	38,1	35,3	29,2	
Maximum Gap Width [m]	0,10	0,30	0,35	0,40	0,40	0,40	0,40	0,40	0,40	0,40	0,40	0,40	0,40	5 Nm
Percentage of Body Length [%]	20,2	44,1	40,3	37,9	32,2	28,0	24,8	22,2	20,1	18,4	16,9	15,7	14,6	

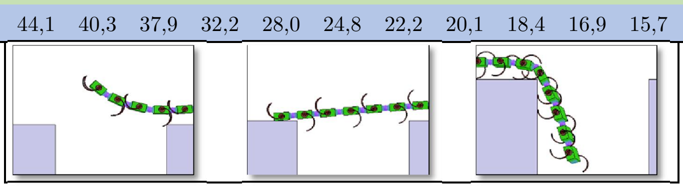


Table 4-1: Results from the gap obstacle simulations with the MCZ with an actively controlled spine, using control method 1 (setting the spine joints to 15°). Two different joint parameter sets are used, one with a 40 Nm maximum torque, and one with a 5 Nm maximum torque.

Table 4-1 and figure 4-1 show the results for the first method to cross a gap with an actively controlled spine. The distance that the robot is able to cross is compared against the length of the robot, both as an absolute distance as well as a percentage of the length of the robot. One thing that immediately became clear while tuning the parameters of the robot, is that the timing of both the spine joints as well as the legs is very important. To maximize the distance that the robot can reach, the legs need to touch down right on the edge of the first platform. A few centimetres off could results in a difference of up to 10 centimetres in the distance that the robot is able to cross. The effect of this can be seen in the difference between the robots with 26, 28, and 30 legs. While the robots with 26 and 28 legs were both able to cross a gap of 0.90 m, the timing of the 30-legged robot didn't match up perfectly with the edge of the platform,

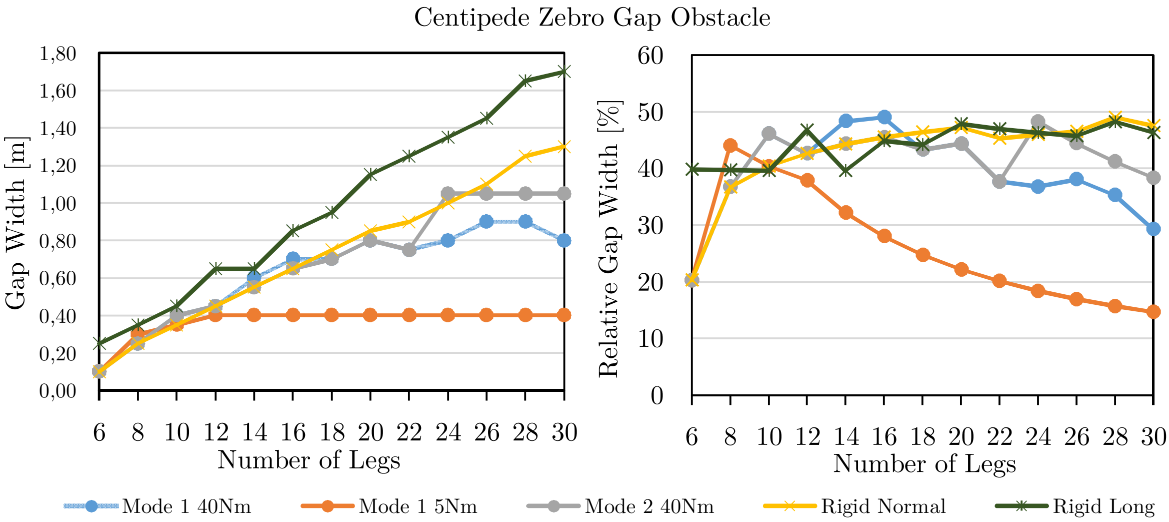


Figure 4-1: Results from the gap obstacle simulations for all the MCZ versions. One the left, the absolute gap width is shown, while on the right the gap width is shown relative to the length of the robots.

resulting in a maximum gap width of 0.80 m. Continuing the simulations by shortening the robot further, showed two main reasons why the distance between the platforms had to be decreased. Either the robot was not able to reach the second platform, or it did reach the second platform but was unable to climb up. The first situation occurs when the weight of the pitched up modules creates a large torque on the last spine joint that is still on the ground (the pivot point is just behind the last legs on the first platform). This torque is counteracted by the other modules that are still on the first platform. When the robot is shortened, this counter torque decreases, making the robot bend at this point. Removing modules at the back of the robot results in the pitched up modules in the front to fall down. Decreasing the distance between the platforms enables the robot to cross again. The second reason that the distance had to be decreased, is the robot being unable to pull itself up when it leaves the first platform. When the last legs move over the edge of the first platform, the back of the robot falls down. Then the front part, which is already on the second platform, has to pull up these modules. If not enough modules are on the second platform at this point, the robot cannot create enough grip and will not be able to climb up. Changing the gait from a synchronous gait to an asynchronous gait improves the traction, but still the ratio between the modules on the second platform and the modules dangling in the gap is important (this this will be called the **climbing ratio** from here on). Experimenting with this climbing ratio has shown that 65% of the modules have to be on the platform before the robot is able to pull the remaining modules up. The effect of this climbing ratio is seen in table 4-1 with the 20 and 22-legged robots. Where the 22-legged robot was able to cross a gap of 0.75 m, this distance could be increased to 0.80 m for the 20-legged robot. The latter gave a better climbing ratio because of the shorter length. When the length of the robot is decreased to 14 legs or less, the spine control scheme becomes less useful. At this point, the distance that the robot can reach decreases fast when the robot is shortened. The most important reason for the robot failing to cross the gap, is that one of the legs gets stuck in the gap, making the robot unstable and causing it to fall in between the platforms. On average, the robot was able to cross a gap of 39% of its length.

The second parameter set used for the spine joints limits the maximum torque to 5 Nm. With this limited torque, the robot is unable to lift up more than just the first module. This inhibits the robot from climbing large gaps, as can be seen in table 4-1. The robot performs the same as the 40 Nm parameter set, up until a length of 12 legs. Increasing the length further does not enhance its ability to cross a gap. The maximum width that this robot can cross is limited to 0.40 m. This results in an average gap width of only 25% of its length.

Active Pitch Control Centipede Zebro Mode 2

Number of Legs	6	8	10	12	14	16	18	20	22	24	26	28	30
Body Length [m]	0,49	0,68	0,87	1,06	1,24	1,43	1,62	1,80	1,99	2,18	2,36	2,55	2,74
Maximum Gap Width [m]	0,10	0,25	0,40	0,45	0,55	0,65	0,70	0,80	0,75	1,05	1,05	1,05	1,05
Percentage of Body Length [%]	20,2	36,7	46,1	42,7	44,3	45,5	43,3	44,4	37,7	48,2	44,4	41,2	38,3

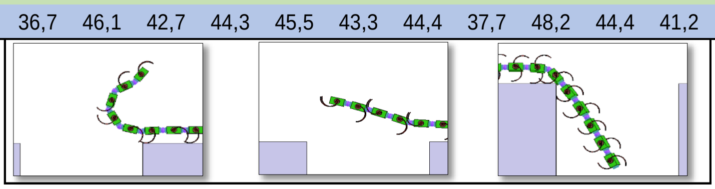


Table 4-2: Results from the gap obstacle simulations with the MCZ with an actively controlled spine, using control method 2 (creating an overhang to decrease the torques on the joints).

A second method for crossing a gap was tested, where the robot folds backwards to create an overhang over the sixth spine joint (more information on this method can be found in section 3-2-1). This method uses a parameter set with a 40 Nm maximum torque. The advantage of this control method is that more modules can be lifted up, increasing the distance that the robot can reach. A greater reach results in an increase of the maximum gap width that the robot is able to cross, as can be seen in table 4-2. The robot is now able to cross a gap of 1.05 m with 24 legs or more. When the length of the robot reaches 22 legs or less, the main problem is the climbing ratio (the ratio between the legs on the second platform versus the legs dangling in the gap), as was also seen with the first method. From here on, the performances of method 1 and 2 are similar. On average, the robot was able to cross a gap of 41% of its length, which is a small improvement over method 1. A 5 Nm maximum torque parameter set was also tried with the second method, but since the robot was not able to lift more than just the first module, the results were exactly the same as with method 1.

	Rigid Centipede Zebro														
Number of Legs	6	8	10	12	14	16	18	20	22	24	26	28	30		
Body Length [m]	0,49	0,68	0,87	1,06	1,24	1,43	1,62	1,80	1,99	2,18	2,36	2,55	2,74	Normal	Spine Size
Maximum Gap Width [m]	0,10	0,25	0,35	0,45	0,55	0,65	0,75	0,85	0,90	1,00	1,10	1,25	1,30		
Percentage of Body Length [%]	20,24	36,71	40,32	42,65	44,28	45,49	46,41	47,14	45,23	45,93	46,53	49	47,48		
Body Length [m]	0,63	0,88	1,14	1,39	1,64	1,90	2,15	2,41	2,66	2,91	3,17	3,42	3,68	Long	
Maximum Gap Width [m]	0,25	0,35	0,45	0,65	0,65	0,85	0,95	1,15	1,25	1,35	1,45	1,65	1,70		
Percentage of Body Length [%]	39,8	39,7	39,6	46,8	39,5	44,8	44,1	47,8	47,0	46,3	45,8	48,2	46,2		

Table 4-3: Results from the gap obstacle simulations with the MCZ with a rigid spine, both a standard and a long version.

In addition to the MCZ robots with an actively controlled spine, also two rigid versions were tested on the gap obstacle (a standard length robot and a long version with double the spine length). The rigid robots perform well when crossing a gap, as can be seen in table 4-3. A maximum gap distance of 1.70 m for the robot with a long spine, and 1.30 m for the robot with the standard length spine, is a large improvement over the compliant robots. Relative to the robot length, the difference is less. The standard length robot can bridge 43% of its body length, while the long robot can bridge 44%. However, the rigid robots are not very practical in a real-life scenario. The simulations show that with a rigid robot, especially a long one with many legs, one misstep makes the robot unstable. It moves like a long stick, balancing on a single point in the middle when it is about to leave the first platform. If it is imbalanced, it sways of the second platform and falls in the gap. In comparison, the Flexible Centipede Zebro (FCZ) robots could tolerate more disturbances than the rigid versions.

To summarize, some statements can be made about the gap crossing simulations:

- Increasing the length of the robot also increases the distance it can cross. Depending on the method, the robot is able to cross around 40% of its body length.
- The maximum joint torques in the spine are important. A low torque prevents the robot from lifting up the modules and has a significant influence on the distance it can cross.
- Using a method that decreases the stress in the joints shows a little improvement in the gap crossing ability of the robot.
- A change of gait from synchronous to asynchronous leg movement when climbing on the second platform, enables the robot to pull up more modules.
- The ratio between the number of modules on the second platform and the modules dangling in the gap (the climbing ratio) is important when the robot is pulling itself up. Simulations have shown that around 65% of the modules need to be on the platform before the robot is able to climb up.
- A rigid spine works well in simulations and would probably also work well in real-life for shorter robots, but long rigid robots easily start to sway. A robot with an active flexible spine has a higher tolerance for disturbances.

4-2 Step Obstacle Results

The step obstacle simulations were conducted in the same manner as for the gap obstacle. First, the longest robot with 30 legs is simulated. The controller parameters are adjusted manually, until it can be said with confidence that the maximum step height that the robot can conquer is found. The robot walks with a synchronous gait against the wall, while the spine follows an s-shape (more on this actuation method can be found in section 3-2). When the last legs leave the ground, the gait is changed to an asynchronous gait, while the robot pulls itself up. For the next simulation, the length of the robot is decreased and the controller parameters are adjusted. Then the step height is decreased until the robot is able to climb up again. With a long body, the main issue is the stability of the robot when it is fully upright and about to climb on the platform. The movement of the legs together with the top-heavy configuration of the robot, makes it fall over to the side easily. A maximum height of 1.20 m can be climbed with the 30-legged robot, as can be seen in table 4-4 and figure 4-2. When the robot becomes shorter, the climbing ratio becomes the main limiting factor, just as can be seen with the gap obstacle. Even though the robot is able to reach the top of the platform, it is unable to pull itself up afterwards if not enough modules are already on the platform. To overcome this problem, a control scheme was tested where the last modules of the robot curl up while climbing on the second platform. This worked on the 30 and 28 legged robots, increasing the maximum gap width with 5 cm, but in all other cases the displacement of the centre of gravity created by this movement made the robot fall backwards. For this reason, this method was not further investigated. When the robot gets a lot shorter (14 legs or less), another problem arises. In this case, when the robot is up against the wall, it easily topples over backwards due to the movement of the legs. When the robot has less than 14 legs, it is unable to completely follow the s-shape and can only pitch up its front. On average, with a 40 Nm maximum joint torque, the robot is able to climb a step of 47% of its length. Just as with the gap obstacle, the maximum torque that the joints can create is important. When the maximum torque is limited to only 5 Nm, the maximum height that the robot is able to climb is decreased to only 0.60 m. This is due to the robot being unable to pitch up its modules. On average, limiting the joint torque to 5 Nm results in the robot being able to climb 37% of its length.

To create a complete overview of the possibilities with the different robot morphologies, also two rigid and a passive compliant version were tested on the step obstacle. As can be seen in table 4-5, the MCZ robot with a rigid spine is able to climb a step with a maximum height of 0.25 m, regardless of the number of legs or the length of the spine. The passive compliant 2-degree-of-freedom (DOF) robot is able to perform a little better, reaching 0.55 m with the 14 legged configuration. Increasing the length further results in the robot folding over backwards since there is no mechanism to push the modules forward on the platform. This means that a passive compliant robot is only useful for small steps.

Active Pitch Control Centipede Zebro Step Obstacle

Number of Legs	6	8	10	12	14	16	18	20	22	24	26	28	30	Maximum Joint Torque
Body Length [m]	0,49	0,68	0,87	1,06	1,24	1,43	1,62	1,80	1,99	2,18	2,36	2,55	2,74	
Maximum Obstacle Height [m]	0,20	0,35	0,35	0,55	0,65	0,65	0,85	0,85	0,85	1,05	1,10	1,10	1,20	40 Nm
Percentage of Body Length [%]	40,5	51,4	40,3	52,1	52,3	45,5	52,6	47,1	42,7	48,2	46,5	43,1	43,8	
Maximum Obstacle Height [m]	0,2	0,35	0,35	0,5	0,55	0,6	0,6	0,6	0,6	0,6	0,6	0,6	0,6	5 Nm
Percentage of Body Length [%]	40,5	51,4	40,3	47,4	44,3	42,0	37,1	33,3	30,2	27,6	25,4	23,5	21,9	

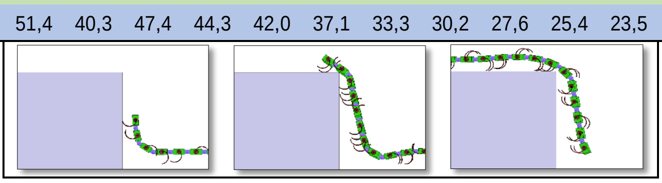


Table 4-4: Results from the step obstacle simulations with the MCZ with an actively controlled spine. Two different joint parameter sets are used, one with a 40 Nm maximum torque and one with a 5 Nm maximum torque.

	Step Height [m]		
# Legs	Rigid Long	Rigid Normal	FCZ 2 DOF
14	0,25	0,25	0,55
12	0,25	0,25	0,35
10	0,25	0,25	0,35
8	0,25	0,25	0,3
6	0,25	0,25	0,2

Table 4-5: Results from the step obstacle simulations with the MCZ with a rigid spine (both a standard length and an extra long spine), and with a passive compliant spine.

The following observations are made from the step obstacle simulations:

- In a long configuration, the robot is top-heavy and can fall over to the side while climbing.
- Just as with the gap obstacle, the ratio between the number of modules on the platform and the modules dangling down is important when the robot is pulling itself up.
- Using active control to curl up the back of the robot while climbing on the second platform can improve the maximum number of modules that it is able to pull up, but more often the displacement of the centre of gravity pulls the robot back.
- Limiting the maximum joint torque has a large influence on the maximum height that the robot is able to climb.
- A rigid robot is able to climb only a small step, and the length of the robot has no influence on its ability to climb.
- A passive compliant robot can climb a step, but will fall backwards when its length increases.

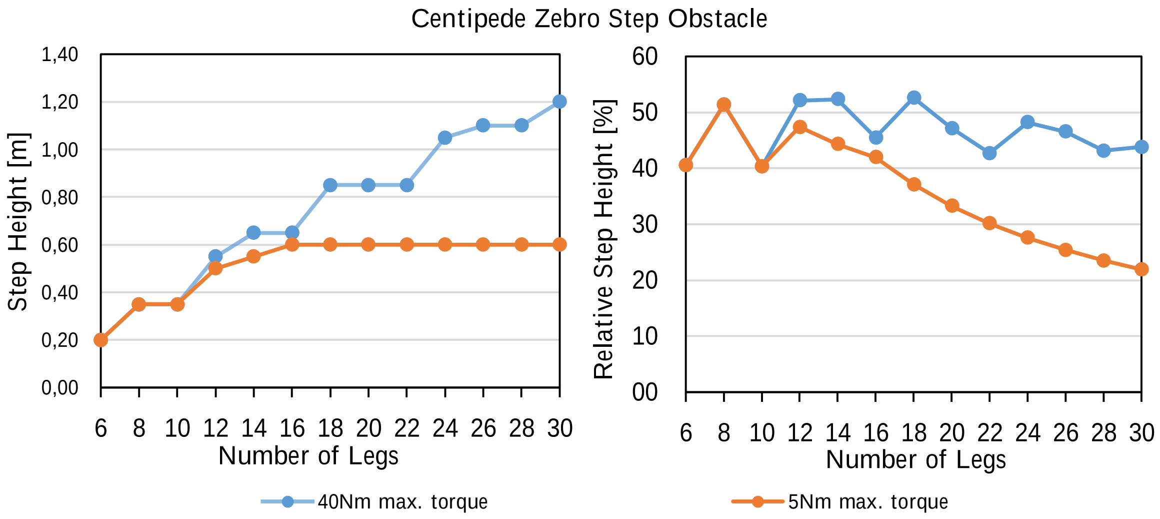


Figure 4-2: Results from the step obstacle simulations for the two parameters sets (40 Nm and 5 Nm maximum torque) with an actively controlled spine on the MCZ. On the left, the absolute step height is shown. On the right, the step height is shown relative to the length of the robots.

4-3 Fractal Landscape Results

The fractal landscape simulations resulted in a lot of data, which was processed as described in the beginning of this chapter. Results from the 8 different versions of the MCZ robot are all presented in separate tables and graphs. Together with 5 tables and 5 graphs for a comparison between robots with the same length, this resulted in 26 tables and graphs in total to present all the information that was gathered during the simulations. Showing them all in this section would be confusing and unclear. For this reason, it was chosen to only present the results for the individual Rigid Centipede Zebro, and the comparison between all the 6-legged robots in this section. The rest of the tables and graphs can be found in appendix A.

The performance of the robot is measured by the distance it can travel over the fractal landscape in a time period of 60 seconds. It is expected that when the terrain becomes more uneven (a larger deviation between the blocks), the robot will travel less far. Also, when the terrain gets rougher, the robot is expected to get stuck at different places, resulting in a larger deviation in the results. Figure 4-3 shows the results for the Rigid Centipede Zebro (RCZ) with different lengths. As was expected, the distance that the robot travels goes down when the average terrain height increases (which also increases the deviation between the blocks that form the terrain). While on a flat landscape the 6-legged robot is able to travel 15.7 m on average, it will only reach 0.85 m on the most difficult terrain. The standard deviation of the travelled distance is fairly large on the easy terrain, but it goes down when the terrain gets rougher. This is counter-intuitive, since a rougher terrain was expected to results in a larger standard deviation. Viewing the simulations revealed that the 6-legged robot easily flips over. Its short body in combination with the powerful and fast legs makes the robot jump and flip when it hits a small obstacle in a certain way. This can happen at any point in the terrain, resulting in a large standard deviation. When the terrain gets rougher, the robot gets

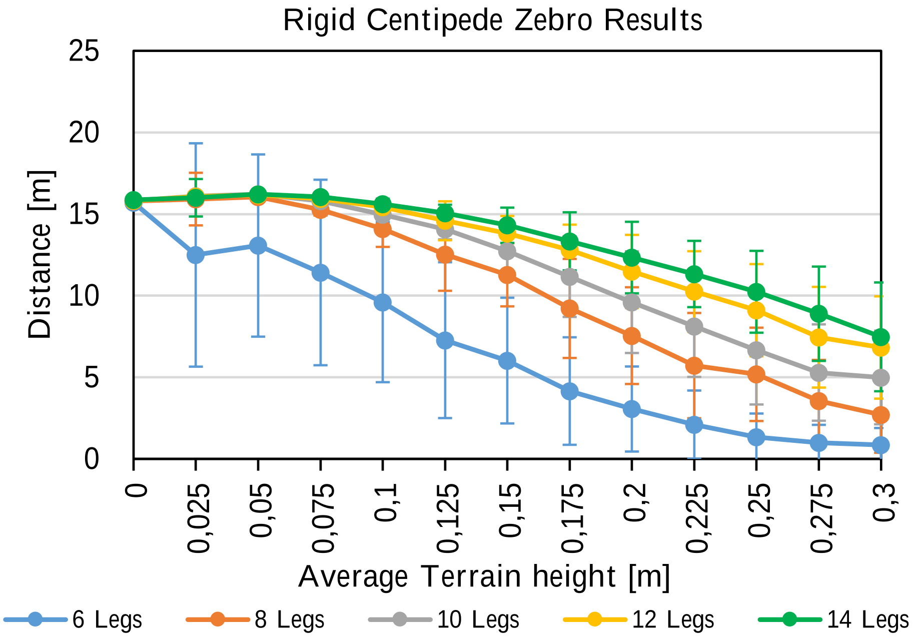


Figure 4-3: Results for the Rigid Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

stuck earlier in the terrain, reducing the spread of the end positions. This results in a smaller standard deviation. Some other versions of the 6-legged MCZ robots show a much smaller spread in the results on the easier terrain, like the Long RCZ, and both passive 2-DOF FCZ versions. The longer spine of the Long RCZ increases its stability, making it less prone to flip over. The FCZ versions are able to absorb some of the shocks that the legs create on the terrain, also preventing it to flip. This explains the smaller standard deviations. Increasing the number of legs on the RCZ also decreases the standard deviation, while increasing the distance that it can travel. A longer body with more legs gives more stability and more grip, resulting in a better performance. While the 8-legged robot seems to perform a lot better than the 6-legged version, the improvement becomes less when the number of legs increases further. The 10-legged robot shows a little improvement, while the 12 and 14-legged versions perform about the same. A similar trend is seen on all 8 different MCZ versions. Increasing the terrain difficulty results in a smaller travelled distance, while increasing the number of legs shows a larger travelled distance. Going from 6 to 8 legs shows more improvement than going from 8 to 10 and further, while the difference between 12 and 14 legs is very small. An one-way ANOVA was performed between consecutive robot lengths within a group of the same robot versions (8 legs compared to 6 legs, 10 legs compared to 8 legs, etc, all on the same MCZ versions). For each of the 8 different MCZ versions, significant differences were found between the different lengths on almost all terrain difficulties. [1]

[1] The one-way Analysis of Variance (ANOVA) test presumes that the data sets represent a normal distribution. A Lilliefors normality test was performed on all data sets, which showed evidence for a normal distribution on some, but not all data sets. Therefore it cannot be guaranteed that all data sets represent a normal distribution. This means that the ANOVA test could be biased. A significance level of $p = 0.05$ was used to discard the null-hypothesis that all samples in the data sets are drawn from populations with the same

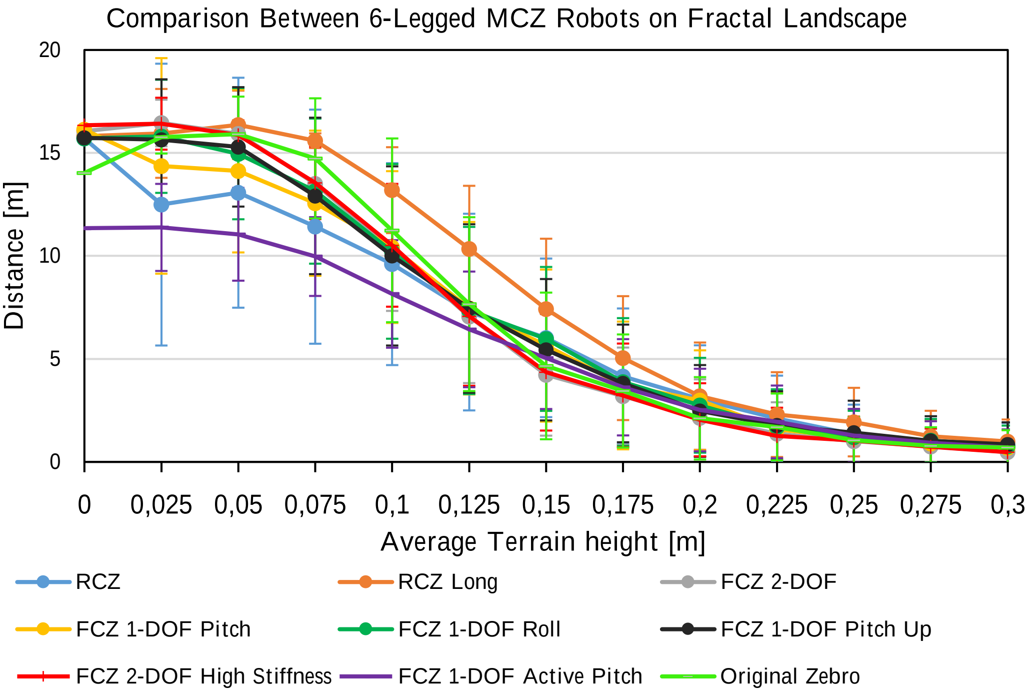


Figure 4-4: Comparison between the 8 different 6-legged MCZ versions.

Figure 4-4 shows a comparison between the 8 different 6-legged MCZ versions. All robots follow a similar curve, with some differences in performance between different robot versions. The Long RCZ performs the best on most terrain difficulties, while the FCZ with an actively controlled spine performs the worst. These differences are most clear on low terrain difficulties. The 6-legged version of the standard RCZ also shows a poor performance compared to the other robots. This changes when the number of legs increases. When the robots are longer, the differences in performance are larger on the high terrain difficulties than on the low difficulties. The RCZ with a standard length spine and the limited pitch FCZ perform best when the length is set to 10 legs or longer. Both the high and standard stiffness 2-DOF FCZ versions show a poor performance for all lengths. The 1-DOF active pitch FCZ robot performs worst with all lengths. Viewing the simulations revealed that the flexible robots curl over the terrain, while the rigid robots move more over the top of the terrain. This means that the flexible robots create more friction and lose speed when folding over the terrain, which decreases their performance. The active pitch FCZ shortens when all spine joint angles are in their maximum position, and stretches again when they are back at 0°. This movement slows down the robot, resulting in a smaller final distance.

After reviewing all the results, some statements can be made about the simulations of the 8 different MCZ versions on the fractal landscape:

- The short 6-legged robots easily flip over, which results in a large deviation in the results. Increasing the body length for stability, or using passive compliance as shock

mean.

absorption, prevents the robot from flipping over and decreases the standard deviation between results.

- Increasing the number of legs shows a better performance. The performance increase between consecutive robot lengths becomes less when the robot becomes longer.
- While the RCZ with a standard spine length performs poorly with a length of 6 legs, it shows a good performance with 10 or more legs.
- The FCZ with active spine control shows the worst overall performance.
- There is almost no difference in the performance between the high and standard stiffness 2-DOF FCZ versions and they both perform poorly compared to the other robots.

In chapter 5 these results will be discussed further so a conclusion can be formed.

Chapter 5

Conclusion & Discussion

Recap of the previous chapters In chapter 1, the concept of the Modular Centipede Zebro (MCZ) was introduced. The purpose of this robot is to navigate through disaster areas and search for victims. In order to do this, the robot must be able to navigate over rough terrain and overcome obstacles like gaps and steps. After researching the body morphologies of several state-of-the-art rough terrain robots, the idea of a modular centipede robot arose. This concept combines the benefits of a swarm of smaller robots to cover a large area fast, with the high rough terrain performance of a long snake-like robot with many legs. The individual swarm-robots are built out of 2-legged modules, with the ability to connect together and form a long chain of modules which moves like a centipede. With this long configuration, the robot should be able overcome obstacles like gaps and steps. The connection between the modules can incorporate one or two degrees-of-freedom (DOFs), both active as well as passive. To test the influence on the performance of the robot by incorporating these DOFs in the spine, as well as changing the length of the robot by adding more modules to the chain, a model of this robot was built in V-REP. More information on this model and the V-REP simulation environment can be found in chapter 2. The control mechanism for the legs and spine is explained in chapter 3. A large number of simulations were conducted to investigate the performances of different body morphologies and lengths on various obstacles. The results of these simulations are revealed in chapter 4. In this last chapter, these results are concluded and discussed.

5-1 Conclusion

In section 1-5, the main hypothesis of this thesis was introduced. It states that:

> "Introducing compliance in the spine and increasing the number of legs on a centipede robot, improves the locomotion performance over rough terrain."

This hypothesis can be divided into 2 sub-hypotheses. The first sub-hypothesis focuses on the spine and states that:

> "Variable compliance in the spine improves mobility over rough terrain."

Variable compliance should be interpreted as introducing one or two DOFs, either active or passive, with certain stiffness levels and control parameters.
The second sub-hypothesis states that:

> "Increasing the number of legs will improve mobility over rough terrain."

Taking a close look at the results may accept or reject these hypotheses. For the first sub-hypothesis, the results of the fractal landscape are examined. Table A-17 through A-25 and graph A-18 through A-26 provide a comparison between the different spine configurations for robots with the same number of legs. First, is must be noted that the 1-degree-of-freedom (DOF) Flexible Centipede Zebro (FCZ) with an actively controlled spine shows the worst performance on almost all terrain difficulties, and for each number of legs. Therefore, it can be concluded that using feed forward control to actuate the spine in a way as was done for this experiment, is not recommended. It will significantly decrease the distance that the robot can reach in a fixed time period on the fractal landscape. To clarify the conclusion for the fractal landscape, the actively controlled FCZ will be excluded from the comparison. For the 6-legged robots, the difference in performance is most evident on the easier terrain. The Rigid Centipede Zebro (RCZ) with the standard length spine shows the second worst performance. Because of its short body, the rigid robot easily flips over. Increasing the length of the spine, as was done with the long RCZ, prevents the robot from flipping over and shows a better performance on the easy terrain. Adding passive DOFs, as for example with the 2-DOF FCZ, also increases the performance, but only on the first 4 terrain difficulties and only with the 6-legged robots. When extra modules are added to the robot, the results are different. On the easy terrain, all robots show a similar performance. When the terrain gets more difficult, differences in performance become more evident. On the high difficulty levels, the RCZ gives the best overall performance, with the 1-DOF limited pitch FCZ as a runner-up. Adding DOFs in the mechanical design of a robot increases its complexity and building cost, while it only shows improvements on the low terrain difficulties and only for the 6-legged configuration. Therefore, it can be concluded that for the fractal landscape (which is the representation for uneven terrain), the first sub-hypothesis can be rejected. In other words; ***adding DOFs to the mechanical design of a modular centipede robot will, in general, not result in a better performance on uneven terrain.***

While this is true for the fractal landscape, the results for the step and gap obstacles show the opposite. The rigid models of the MCZ showed a good performance in simulations on the gap obstacle, however these would probably not perform well in a real-life scenario due to a low tolerance for disturbances. To enable the robot to cross a large gap or climb a high step, it is necessary to incorporate an actively controlled pitch motion in the spine. Also, the maximum torque that the joints can provide is important. Compared to a 40 Nm maximum spine torque, the 5 Nm maximum spine torque simulations showed a significant reduction in performance. In conclusion, for the step and gap obstacles, the first sub-hypothesis can be accepted with respect to an actively controlled pitch motion in the spine. This mean that, ***to enable a centipede robot to climb high steps and cross large gaps, an actively controlled pitch motion must be incorporated in the spine, preferably with a high maximum torque.***

Increasing the length of the MCZ robot shows improvements in performance on the fractal landscape, as well as on the gap and step obstacles. For the fractal landscape, the largest improvement is seen when the length is increased from 6 to 8 legs. Adding more modules to give the robot a length of 10 or 12 legs, also increases the performance. However, this increase is less than going from 6 to 8 legs. Making the robot even longer is unnecessary, since the increase in performance is only marginal. Therefore, the second sub-hypothesis is accepted for the fractal landscape. Since adding more modules is also more costly, it can be said that ***for a rigid centipede robot on rough terrain, a length of 8 legs is advised.***

For the gap and step obstacles, increasing the length of the FCZ with an actively controlled pitch motion, shows a large increase in performance. Different robot lengths were tested, ranging from 6 to 30 legs. On average, the robots were able to climb a step of 47% of their length and cross a gap of 40% of their length. Since the robot's length was not increased any further than 30 legs, it cannot be said with certainty that adding more modules will also increase the robots ability to overcome these obstacles. However, viewing the simulations gave the impressions that at 30 legs, the robots were at the limits of their capabilities. With 30 legs, the robots would almost become unstable, so adding more modules would probably show the same results. Therefore, sub-hypothesis 2 can also be accepted for the gap and step obstacles. In other words, ***increasing the length of a centipede robot with an actively controlled spine, will enable it to cross a larger gap or climb a higher step.***

The goal of this thesis project is to provide a first exploration of the possibilities of a modular centipede robot for rough terrain navigation. The results from the numerous simulations, together with the conclusions, can be used as a design guide for such a robot. To summarize, the following observations can be made from the conclusion:

- For uneven terrain, adding DOFs does not results in a better performance and only complicates the design and raises the cost.
- For uneven terrain, using 8 or 10 legs is recommended. Using only 6 legs significantly reduces the performance.
- For climbing steps or crossing gaps, an actively controlled pitch motion in the spine, preferably with a high maximum torque, greatly increases performance.
- With the use of an active pitch motion in the spine, the robot is able to cross a gap of on average 40% of its length. For climbing steps, the robot is able to climb on average 47% of its length.

- There is a limit to the maximum gap width and step height that the robot is able to climb. For instance, a robot with 100 legs would probably not be able to cross a gap of 40% of its length. The same holds for step climbing.

By taking the previous observations into account, a basic first design can be suggested for the MCZ. To be able to overcome the various obstacles found in disaster areas, the robot should incorporate a powerful active pitch motion in the spine. The pitch motion should be locked when the robot is travelling over rough terrain. This can be realized with the use of a mechanical break or locking pin, or by using a non-backdrivable gear mechanism like a worm-wheel. For rough terrain navigation in general, the robot should have 8 or 10 legs. When the robot is confronted with a step or a gap, it should connect together with other robots to form a configuration with the appropriate length to overcome the obstacle.

5-2 Discussion

For this thesis project, the general idea of a modular centipede robot was explored. From a robotics standpoint, such a robot would be a great improvement over the existing robots in the field of Urban Search and Rescue (USAR). The MCZ would be able to cover areas that, up until now, could not be explored by robots in a fast and efficient way. However, from a mechanical engineering standpoint, the proposed modular centipede robot would be extremely difficult to design and build. The spine connection, which is the most important part of the robot, must be able to autonomously connect modules together while standing on uneven terrain. It will be difficult for different robots to navigate towards each other and position themselves with such accuracy, so that the offset between the connecting parts of the spine is so small that they will be able to connect together autonomously. Furthermore, the modules must be small, lightweight, and they must be able to function by themselves. This means that they all need their own power source and computational core. The modules also need to communicate with each other, preferably wireless. To detect the terrain difficulty and measure obstacles, some of the units must be equipped with advanced sensors. The results from this thesis can then be used by the robots to form a strategy to reach their goal. Altogether, the design of the MCZ is a great challenge. However, if such a robot could be designed, it would be a great addition to the equipment at the disposal of search and rescue workers. Hopefully, in the near future, humans and robots will work together and greatly increase the survival chances of victims in disaster areas.

Appendix A

Fractal Landscape Simulation Tables and Graphs

A-1 Individual MCZ Results

The following tables and graphs present the results for each Modular Centipede Zebro (MCZ) version after simulations on the fractal landscape.

Rigid Centipede Zebro

	Average Terrain Height [m]													
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	Number of Legs
D[m]	15,70	12,49	13,07	11,42	9,60	7,27	6,02	4,16	3,06	2,10	1,34	0,99	0,85	6
σ[m]	0,00	6,84	5,58	5,68	4,90	4,77	3,85	3,29	2,61	2,09	1,44	1,09	1,04	
D[m]	15,81	15,92	16,05	15,27	14,09	12,53	11,28	9,22	7,55	5,72	5,18	3,54	2,70	8
σ[m]	0,00	1,61	0,22	0,36	1,10	2,23	1,93	3,03	2,96	3,22	2,86	2,53	2,33	
D[m]	15,83	16,10	16,22	15,79	14,98	14,07	12,72	11,16	9,60	8,11	6,65	5,29	4,99	10
σ[m]	0,00	0,17	0,19	0,27	0,64	0,62	1,70	2,47	3,11	3,09	3,32	2,95	2,87	
D[m]	15,83	16,07	16,18	15,93	15,43	14,59	13,82	12,81	11,48	10,25	9,11	7,45	6,82	12
σ[m]	0,00	0,15	0,22	0,27	0,38	1,19	1,07	1,54	2,24	2,47	2,82	3,08	3,13	
D[m]	15,87	16,00	16,22	16,05	15,61	15,06	14,31	13,33	12,33	11,33	10,24	8,89	7,48	14
σ[m]	0,00	1,15	0,22	0,26	0,33	0,51	1,08	1,78	2,19	2,03	2,51	2,89	3,33	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-1: Results for the Rigid Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

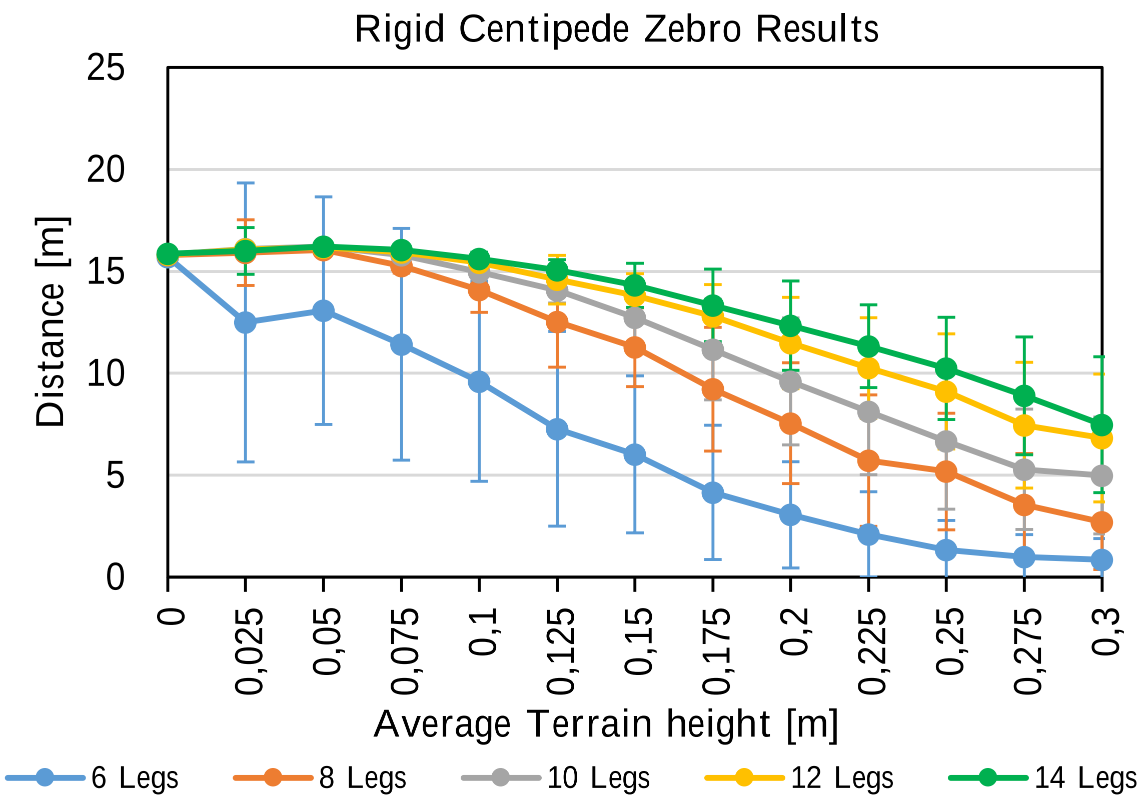


Figure A-2: Results for the Rigid Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

Rigid Long Centipede Zebro

	Average Terrain Height [m]													
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	Number of Legs
D[m]	15,82	15,95	16,37	15,59	13,19	10,34	7,43	5,04	3,19	2,30	1,93	1,24	0,99	6
σ[m]	0,00	2,16	1,66	0,36	2,09	3,07	3,41	3,01	2,60	2,06	1,66	1,24	1,07	
D[m]	15,82	15,90	16,47	15,90	14,64	12,67	10,86	9,19	7,30	5,72	4,87	3,64	2,92	8
σ[m]	0,00	2,28	1,19	1,17	0,68	2,51	2,64	3,03	3,10	3,23	2,86	2,53	2,15	
D[m]	15,88	16,16	16,67	16,43	15,25	13,86	12,10	10,10	8,20	6,15	4,83	3,69	3,17	10
σ[m]	0,00	1,22	0,17	0,28	1,91	1,99	2,65	3,21	3,39	3,52	3,24	2,61	2,56	
D[m]	15,86	16,18	16,64	16,52	15,95	14,84	13,35	12,32	10,46	8,78	8,00	6,33	5,34	12
σ[m]	0,00	0,12	0,18	1,20	0,47	1,49	2,19	1,95	2,95	3,32	3,14	3,15	3,05	
D[m]	15,89	16,09	16,61	16,71	16,21	15,38	14,28	12,75	11,00	9,77	8,07	7,08	6,10	14
σ[m]	0,00	1,15	0,17	0,26	0,40	0,54	1,38	2,23	3,17	3,40	3,44	3,60	3,46	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-3: Results for the Rigid Centipede Zebro with an extra long spine on the fractal landscape, lengths range from 6 legs to 14 legs.

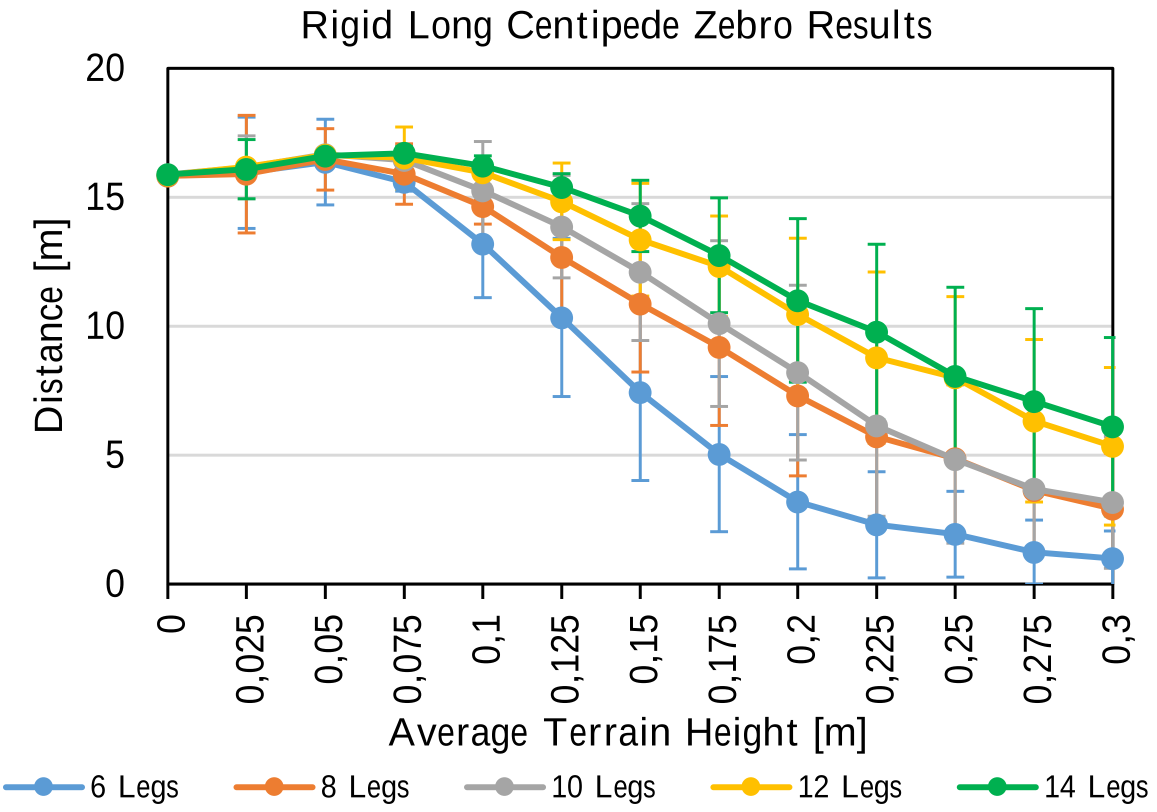


Figure A-4: Results for the Rigid Centipede Zebro with an extra long spine on the fractal landscape, lengths range from 6 legs to 14 legs.

Flexible 2-DOF Pitch & Roll Centipede Zebro

	Average Terrain Height [m]													
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	Number of Legs
D[m]	16,06	16,45	15,93	13,51	10,32	7,06	4,23	3,19	2,14	1,38	1,01	0,75	0,48	6
σ[m]	0,00	1,13	0,58	2,12	2,99	3,23	2,95	2,36	1,87	1,52	1,10	0,88	0,68	
D[m]	16,39	16,68	16,47	15,29	13,46	11,10	9,08	7,28	4,98	3,53	3,12	2,03	1,63	8
σ[m]	0,00	0,14	0,24	0,36	0,81	2,34	2,54	2,75	2,76	2,37	2,03	1,73	1,57	
D[m]	15,99	16,48	16,29	15,32	14,04	12,22	10,46	8,74	6,84	5,39	4,51	3,39	2,87	10
σ[m]	0,00	0,16	0,24	0,69	1,15	2,26	2,42	2,95	3,16	2,98	2,68	2,26	2,06	
D[m]	16,16	16,48	16,48	15,79	14,63	13,32	11,69	9,95	8,57	6,83	5,96	4,46	3,86	12
σ[m]	0,00	0,19	0,19	0,34	0,61	1,28	1,87	2,61	2,92	3,01	2,84	2,56	2,58	
D[m]	16,35	16,43	16,23	15,48	14,36	12,75	11,39	9,73	8,10	6,98	6,06	4,65	4,30	14
σ[m]	0,00	0,17	0,21	0,31	0,55	1,64	1,85	2,44	3,06	2,72	2,77	2,59	2,55	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-5: Results for the flexible 2-DOF Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

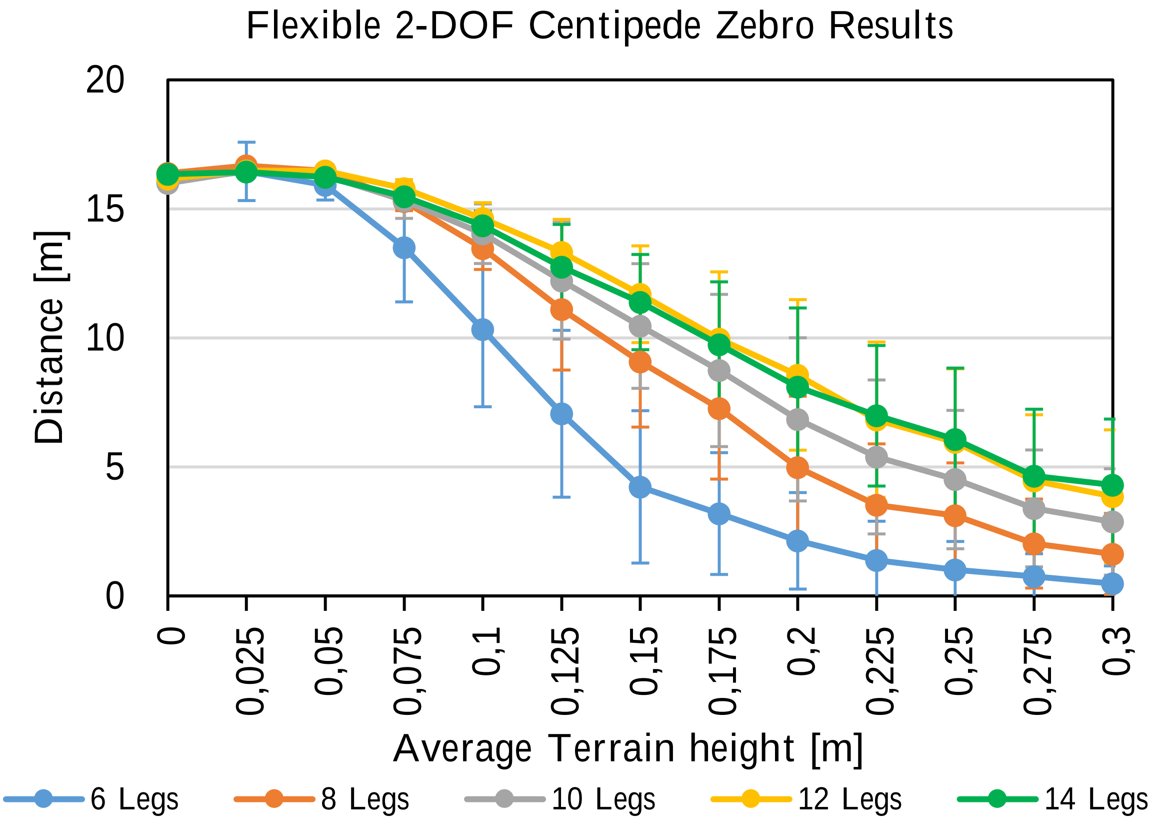


Figure A-6: Results for the flexible 2-DOF Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

Flexible 1-DOF Pitch Centipede Zebro

	Average Terrain Height [m]													
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	Number of Legs
D[m]	16,13	14,37	14,12	12,56	10,43	7,46	5,65	3,71	2,98	1,58	1,18	0,90	0,68	6
σ[m]	0,00	5,23	3,95	3,52	3,68	4,19	3,69	3,10	2,43	1,77	1,35	1,09	0,86	
D[m]	16,09	16,07	15,97	15,14	13,92	12,52	11,22	9,19	7,35	5,90	4,80	3,37	2,58	8
σ[m]	0,00	0,20	0,23	0,34	0,49	1,00	1,72	2,61	2,82	2,97	2,82	2,61	2,19	
D[m]	16,01	16,25	16,16	15,55	14,73	13,44	12,23	10,53	8,95	7,48	6,40	4,90	3,97	10
σ[m]	0,00	0,16	0,23	0,31	0,40	1,49	1,75	2,57	3,05	3,00	3,02	2,80	2,66	
D[m]	16,17	16,33	16,34	15,84	14,62	13,15	11,41	9,94	8,39	6,90	6,34	4,78	4,71	12
σ[m]	0,00	0,15	0,21	0,29	1,54	2,24	2,66	3,00	3,33	3,25	2,86	2,77	2,62	
D[m]	16,11	16,22	16,26	15,71	14,63	13,08	11,51	9,95	7,92	6,64	5,89	4,90	4,09	14
σ[m]	0,00	0,15	0,21	0,33	0,73	1,57	2,41	2,67	3,36	3,01	3,04	2,71	2,57	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-7: Results for the flexible 1-DOF pitch Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

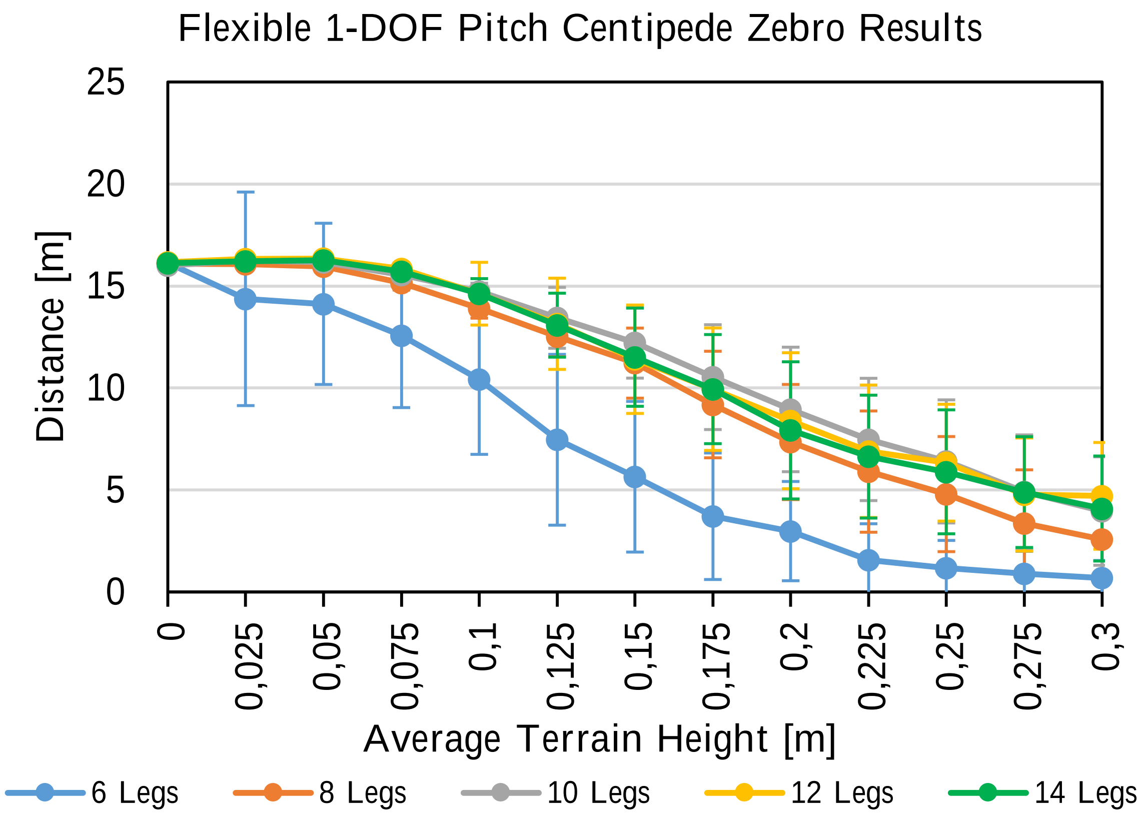


Figure A-8: Results for the flexible 1-DOF pitch Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

Flexible 1-DOF Roll Centipede Zebro

	Average Terrain Height [m]													Number of Legs
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	
D[m]	15,71	15,81	14,96	13,15	10,22	7,35	5,97	3,88	2,75	1,69	1,19	1,03	0,75	6
σ[m]	0,00	2,75	3,18	3,52	4,24	4,06	3,49	3,10	2,30	1,83	1,29	1,05	1,01	
D[m]	16,11	16,34	16,30	15,34	13,90	12,14	10,52	8,53	6,80	4,94	3,83	2,72	2,17	8
σ[m]	0,00	0,22	0,21	0,38	0,84	1,93	1,84	2,68	2,86	2,83	2,61	2,29	1,93	
D[m]	16,03	16,47	16,56	15,82	14,57	12,87	11,24	8,93	7,54	5,93	5,28	3,82	3,30	10
σ[m]	0,00	0,16	0,20	0,30	0,56	1,49	2,22	3,15	3,11	3,00	2,68	2,58	2,25	
D[m]	16,19	16,56	16,52	15,90	14,75	13,34	11,83	10,08	8,59	7,03	6,19	4,45	3,92	12
σ[m]	0,00	0,15	0,19	0,29	0,63	1,32	1,45	2,39	2,68	2,81	2,75	2,62	2,58	
D[m]	16,25	16,64	16,53	15,98	14,90	13,82	12,32	10,80	9,38	7,51	6,99	5,47	4,89	14
σ[m]	0,00	0,16	0,23	0,30	0,72	0,84	1,71	2,34	2,60	2,84	2,75	2,56	2,53	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-9: Results for the flexible 1-DOF roll Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

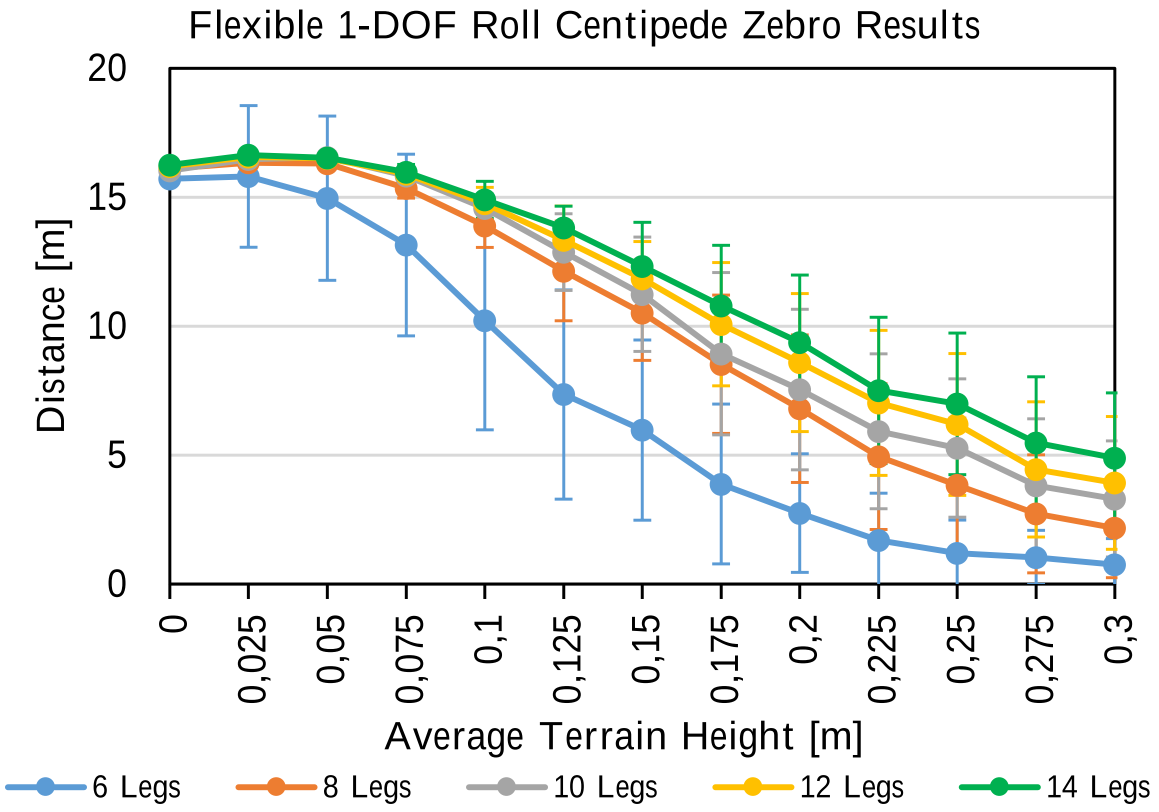
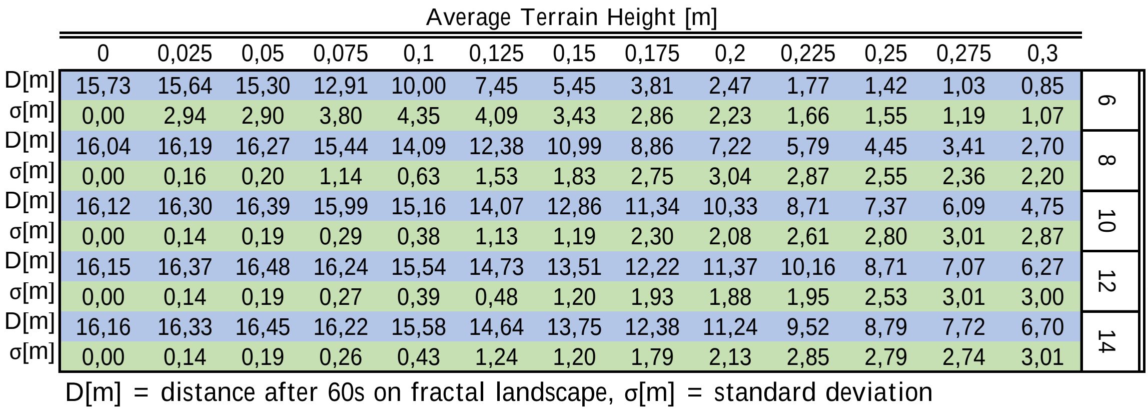


Figure A-10: Results for the flexible 1-DOF roll Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

Flexible 1-DOF Limited Pitch Centipede Zebro



Average Terrain Height [m]

	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	Number of Legs
D[m]	15,73	15,64	15,30	12,91	10,00	7,45	5,45	3,81	2,47	1,77	1,42	1,03	0,85	6
σ[m]	0,00	2,94	2,90	3,80	4,35	4,09	3,43	2,86	2,23	1,66	1,55	1,19	1,07	
D[m]	16,04	16,19	16,27	15,44	14,09	12,38	10,99	8,86	7,22	5,79	4,45	3,41	2,70	8
σ[m]	0,00	0,16	0,20	1,14	0,63	1,53	1,83	2,75	3,04	2,87	2,55	2,36	2,20	
D[m]	16,12	16,30	16,39	15,99	15,16	14,07	12,86	11,34	10,33	8,71	7,37	6,09	4,75	10
σ[m]	0,00	0,14	0,19	0,29	0,38	1,13	1,19	2,30	2,08	2,61	2,80	3,01	2,87	
D[m]	16,15	16,37	16,48	16,24	15,54	14,73	13,51	12,22	11,37	10,16	8,71	7,07	6,27	12
σ[m]	0,00	0,14	0,19	0,27	0,39	0,48	1,20	1,93	1,88	1,95	2,53	3,01	3,00	
D[m]	16,16	16,33	16,45	16,22	15,58	14,64	13,75	12,38	11,24	9,52	8,79	7,72	6,70	14
σ[m]	0,00	0,14	0,19	0,26	0,43	1,24	1,20	1,79	2,13	2,85	2,79	2,74	3,01	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-11: Results for the flexible 1-DOF Limited Pitch Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

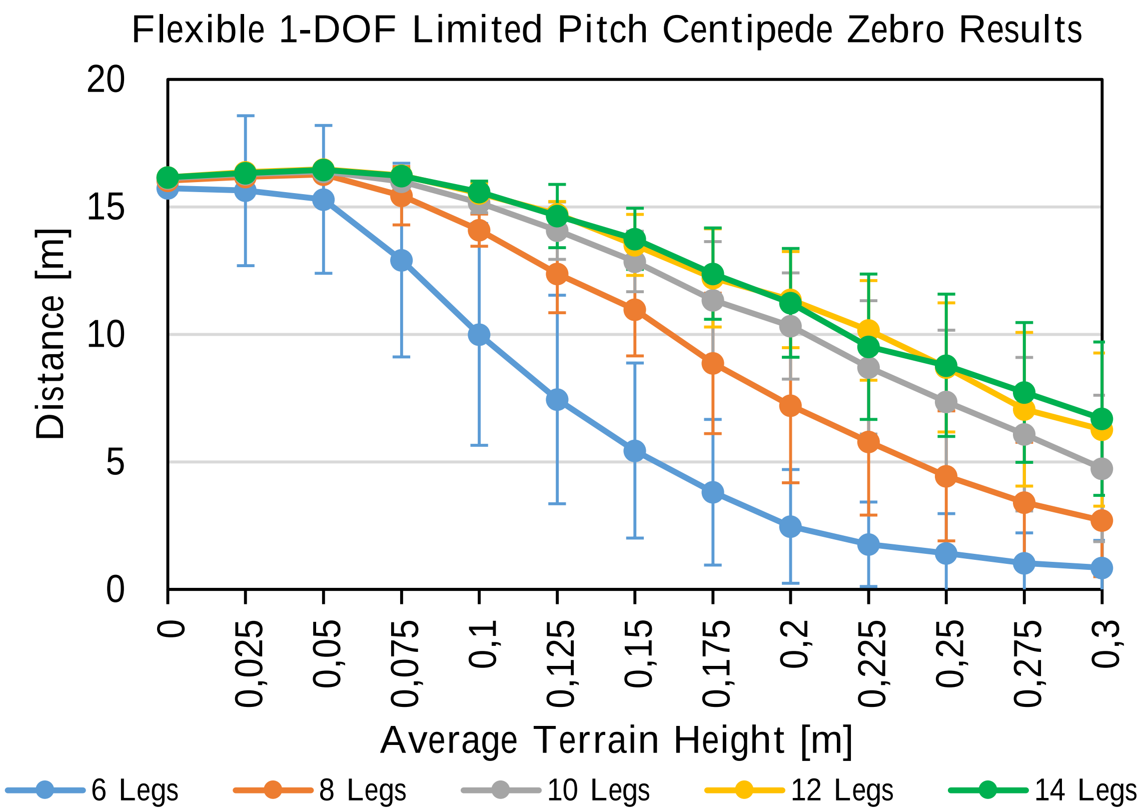


Figure A-12: Results for the flexible 1-DOF Limited Pitch Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

Flexible 2-DOF High Stiffness Centipede Zebro

	Average Terrain Height [m]													
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	Number of Legs
D[m]	16,35	16,42	15,87	13,55	10,52	7,08	4,36	3,23	2,05	1,26	1,04	0,73	0,47	6
σ[m]	0,00	1,26	0,71	1,76	2,98	3,38	2,83	2,51	1,77	1,37	1,16	0,88	0,61	
D[m]	16,41	16,67	16,50	15,28	13,47	11,33	9,32	6,96	5,29	4,12	3,08	2,20	1,89	8
σ[m]	0,00	0,15	0,21	0,40	1,05	1,71	2,32	2,85	2,58	2,38	2,14	1,78	1,55	
D[m]	16,14	16,44	16,27	15,41	13,98	12,22	10,34	8,15	6,93	5,86	4,31	3,67	2,86	10
σ[m]	0,00	0,18	0,22	0,35	1,44	2,34	2,93	3,53	3,01	2,87	2,64	2,49	2,03	
D[m]	16,14	16,48	16,43	15,79	14,74	13,34	11,62	10,18	8,49	7,58	5,58	4,89	3,88	12
σ[m]	0,00	0,18	0,20	0,29	0,49	1,33	2,14	2,47	2,71	2,62	2,86	2,87	2,41	
D[m]	16,29	16,44	16,21	15,52	14,38	12,94	11,38	9,78	8,32	7,05	5,79	4,70	4,33	14
σ[m]	0,00	0,17	0,23	0,33	0,91	1,20	2,11	2,42	2,83	2,79	2,62	2,73	2,51	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-13: Results for the flexible 2-DOF High Stiffness Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

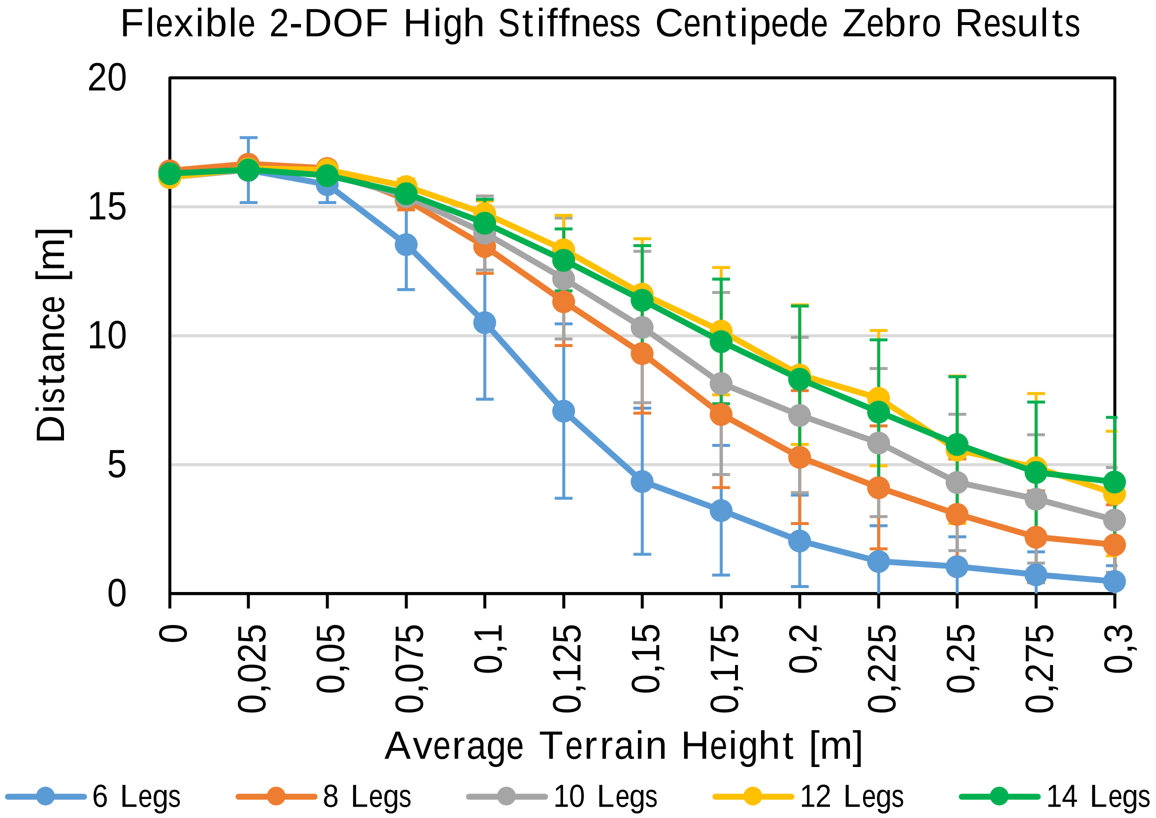
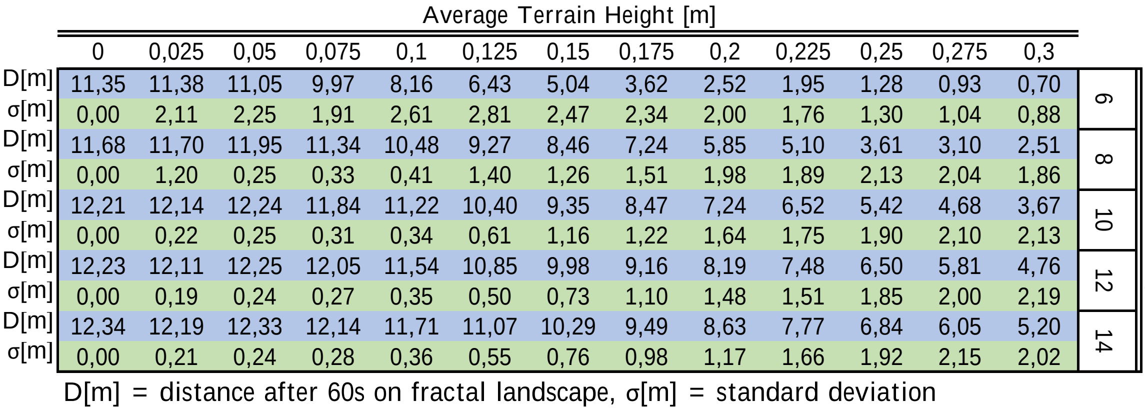


Figure A-14: Results for the flexible 2-DOF High Stiffness Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

Flexible 1-DOF Active Pitch Centipede Zebro



	Average Terrain Height [m]													Number of Legs
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	
D[m]	11,35	11,38	11,05	9,97	8,16	6,43	5,04	3,62	2,52	1,95	1,28	0,93	0,70	6
σ[m]	0,00	2,11	2,25	1,91	2,61	2,81	2,47	2,34	2,00	1,76	1,30	1,04	0,88	
D[m]	11,68	11,70	11,95	11,34	10,48	9,27	8,46	7,24	5,85	5,10	3,61	3,10	2,51	8
σ[m]	0,00	1,20	0,25	0,33	0,41	1,40	1,26	1,51	1,98	1,89	2,13	2,04	1,86	
D[m]	12,21	12,14	12,24	11,84	11,22	10,40	9,35	8,47	7,24	6,52	5,42	4,68	3,67	10
σ[m]	0,00	0,22	0,25	0,31	0,34	0,61	1,16	1,22	1,64	1,75	1,90	2,10	2,13	
D[m]	12,23	12,11	12,25	12,05	11,54	10,85	9,98	9,16	8,19	7,48	6,50	5,81	4,76	12
σ[m]	0,00	0,19	0,24	0,27	0,35	0,50	0,73	1,10	1,48	1,51	1,85	2,00	2,19	
D[m]	12,34	12,19	12,33	12,14	11,71	11,07	10,29	9,49	8,63	7,77	6,84	6,05	5,20	14
σ[m]	0,00	0,21	0,24	0,28	0,36	0,55	0,76	0,98	1,17	1,66	1,92	2,15	2,02	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-15: Results for the flexible 1-DOF Active Pitch Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

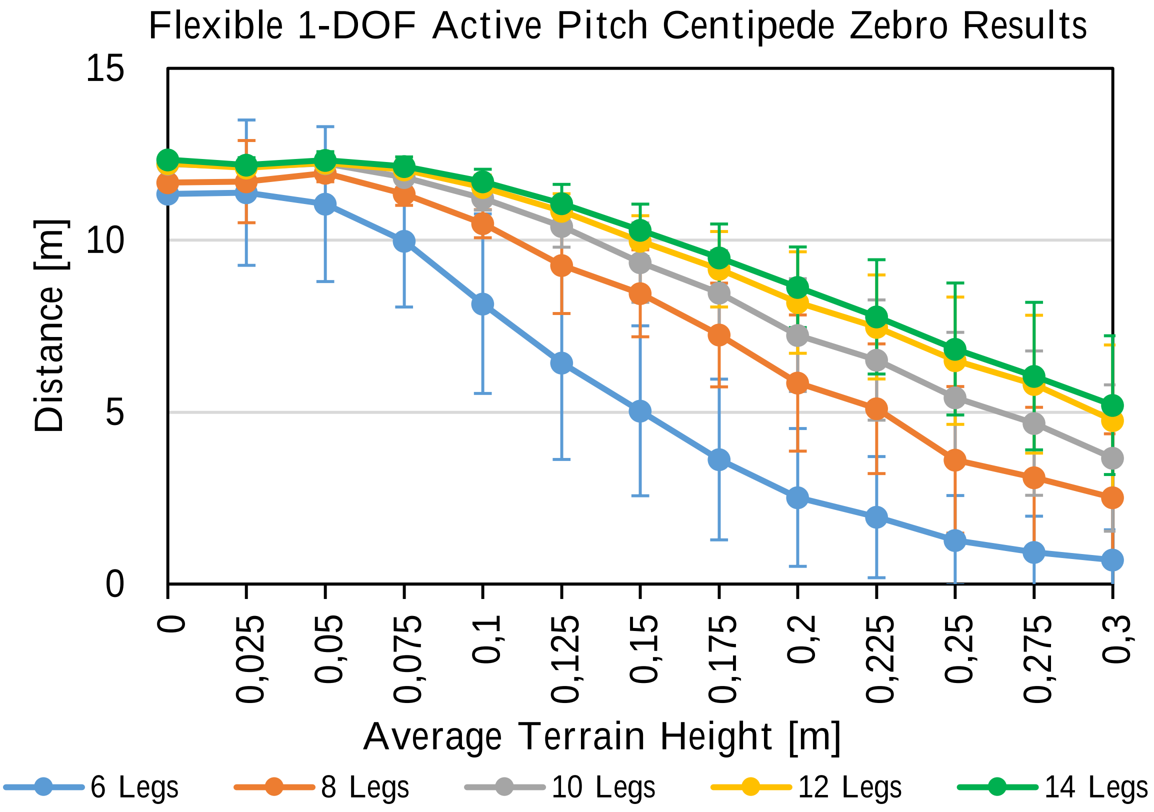


Figure A-16: Results for the flexible 1-DOF Active Pitch Centipede Zebro on the fractal landscape, lengths range from 6 legs to 14 legs.

A-2 Comparison Between Different MCZ Versions

The following tables and graphs provide a comparison between all the different MCZ versions with the same number of legs.

Comparison Between 6-Legged MCZ Robots

	Average Terrain Height [m]													
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	
D[m]	15,70	12,49	13,07	11,42	9,60	7,27	6,02	4,16	3,06	2,10	1,34	0,99	0,85	RCZ
σ[m]	0,00	6,84	5,58	5,68	4,90	4,77	3,85	3,29	2,61	2,09	1,44	1,09	1,04	
D[m]	15,82	15,95	16,37	15,59	13,19	10,34	7,43	5,04	3,19	2,30	1,93	1,24	0,99	RCZ Long
σ[m]	0,00	2,16	1,66	0,36	2,09	3,07	3,41	3,01	2,60	2,06	1,66	1,24	1,07	
D[m]	16,06	16,45	15,93	13,51	10,32	7,06	4,23	3,19	2,14	1,38	1,01	0,75	0,48	FCZ 2-DOF
σ[m]	0,00	1,13	0,58	2,12	2,99	3,23	2,95	2,36	1,87	1,52	1,10	0,88	0,68	
D[m]	16,13	14,37	14,12	12,56	10,43	7,46	5,65	3,71	2,98	1,58	1,18	0,90	0,68	FCZ 1-DOF Pitch
σ[m]	0,00	5,23	3,95	3,52	3,68	4,19	3,69	3,10	2,43	1,77	1,35	1,09	0,86	
D[m]	15,71	15,81	14,96	13,15	10,22	7,35	5,97	3,88	2,75	1,69	1,19	1,03	0,75	FCZ 1-DOF Roll
σ[m]	0,00	2,75	3,18	3,52	4,24	4,06	3,49	3,10	2,30	1,83	1,29	1,05	1,01	
D[m]	15,73	15,64	15,30	12,91	10,00	7,45	5,45	3,81	2,47	1,77	1,42	1,03	0,85	FCZ 1-DOF Pitch Up
σ[m]	0,00	2,94	2,90	3,80	4,35	4,09	3,43	2,86	2,23	1,66	1,55	1,19	1,07	
D[m]	16,35	16,42	15,87	13,55	10,52	7,08	4,36	3,23	2,05	1,26	1,04	0,73	0,47	FCZ 2-DOF High Stiffness
σ[m]	0,00	1,26	0,71	1,76	2,98	3,38	2,83	2,51	1,77	1,37	1,16	0,88	0,61	
D[m]	11,35	11,38	11,05	9,97	8,16	6,43	5,04	3,62	2,52	1,95	1,28	0,93	0,70	FCZ 1-DOF Active Pitch
σ[m]	0,00	2,11	2,25	1,91	2,61	2,81	2,47	2,34	2,00	1,76	1,30	1,04	0,88	
D[m]	14,03	15,77	15,91	14,74	11,24	7,66	4,66	3,44	2,12	1,70	1,06	0,78	0,70	Original Zebro
σ[m]	0,00	0,80	1,82	2,92	4,46	4,22	3,56	2,75	2,00	1,62	1,12	0,90	0,84	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-17: Comparison between the 8 different 6-legged MCZ versions.

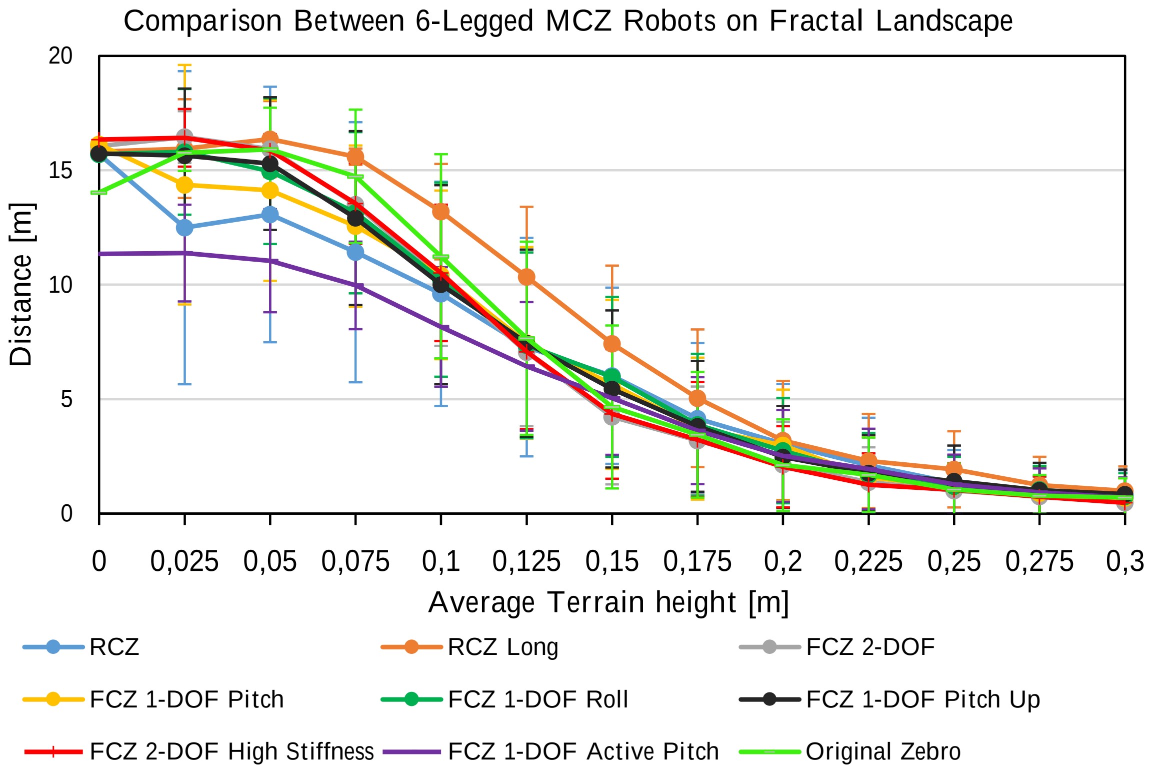
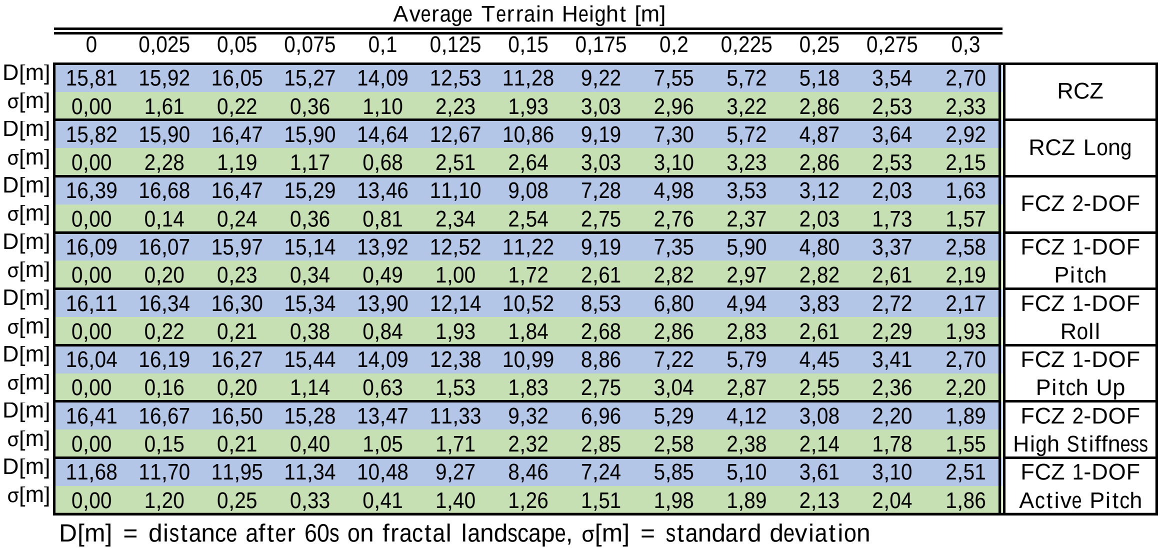


Figure A-18: Comparison between the 8 different 6-legged MCZ versions.

Comparison Between 8-Legged MCZ Robots



Average Terrain Height [m]

	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	
D[m]	15,81	15,92	16,05	15,27	14,09	12,53	11,28	9,22	7,55	5,72	5,18	3,54	2,70	RCZ
σ[m]	0,00	1,61	0,22	0,36	1,10	2,23	1,93	3,03	2,96	3,22	2,86	2,53	2,33	
D[m]	15,82	15,90	16,47	15,90	14,64	12,67	10,86	9,19	7,30	5,72	4,87	3,64	2,92	RCZ Long
σ[m]	0,00	2,28	1,19	1,17	0,68	2,51	2,64	3,03	3,10	3,23	2,86	2,53	2,15	
D[m]	16,39	16,68	16,47	15,29	13,46	11,10	9,08	7,28	4,98	3,53	3,12	2,03	1,63	FCZ 2-DOF
σ[m]	0,00	0,14	0,24	0,36	0,81	2,34	2,54	2,75	2,76	2,37	2,03	1,73	1,57	
D[m]	16,09	16,07	15,97	15,14	13,92	12,52	11,22	9,19	7,35	5,90	4,80	3,37	2,58	FCZ 1-DOF Pitch
σ[m]	0,00	0,20	0,23	0,34	0,49	1,00	1,72	2,61	2,82	2,97	2,82	2,61	2,19	
D[m]	16,11	16,34	16,30	15,34	13,90	12,14	10,52	8,53	6,80	4,94	3,83	2,72	2,17	FCZ 1-DOF Roll
σ[m]	0,00	0,22	0,21	0,38	0,84	1,93	1,84	2,68	2,86	2,83	2,61	2,29	1,93	
D[m]	16,04	16,19	16,27	15,44	14,09	12,38	10,99	8,86	7,22	5,79	4,45	3,41	2,70	FCZ 1-DOF Pitch Up
σ[m]	0,00	0,16	0,20	1,14	0,63	1,53	1,83	2,75	3,04	2,87	2,55	2,36	2,20	
D[m]	16,41	16,67	16,50	15,28	13,47	11,33	9,32	6,96	5,29	4,12	3,08	2,20	1,89	FCZ 2-DOF High Stiffness
σ[m]	0,00	0,15	0,21	0,40	1,05	1,71	2,32	2,85	2,58	2,38	2,14	1,78	1,55	
D[m]	11,68	11,70	11,95	11,34	10,48	9,27	8,46	7,24	5,85	5,10	3,61	3,10	2,51	FCZ 1-DOF Active Pitch
σ[m]	0,00	1,20	0,25	0,33	0,41	1,40	1,26	1,51	1,98	1,89	2,13	2,04	1,86	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-19: Comparison between the 8 different 8-legged MCZ versions.

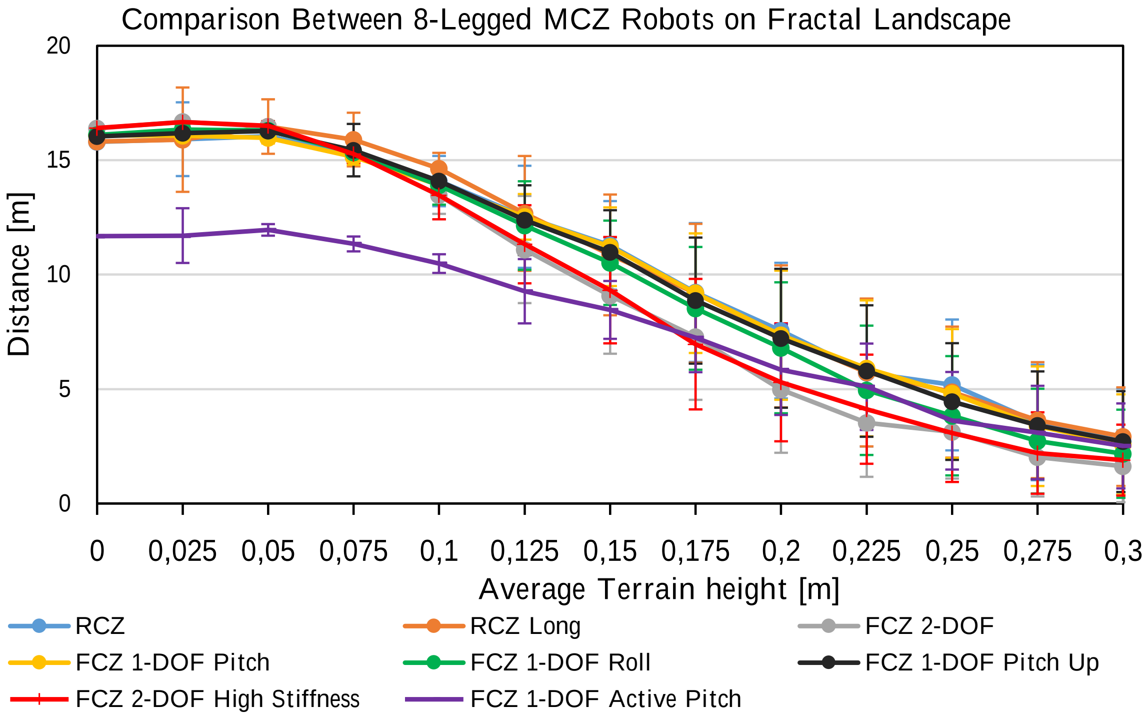
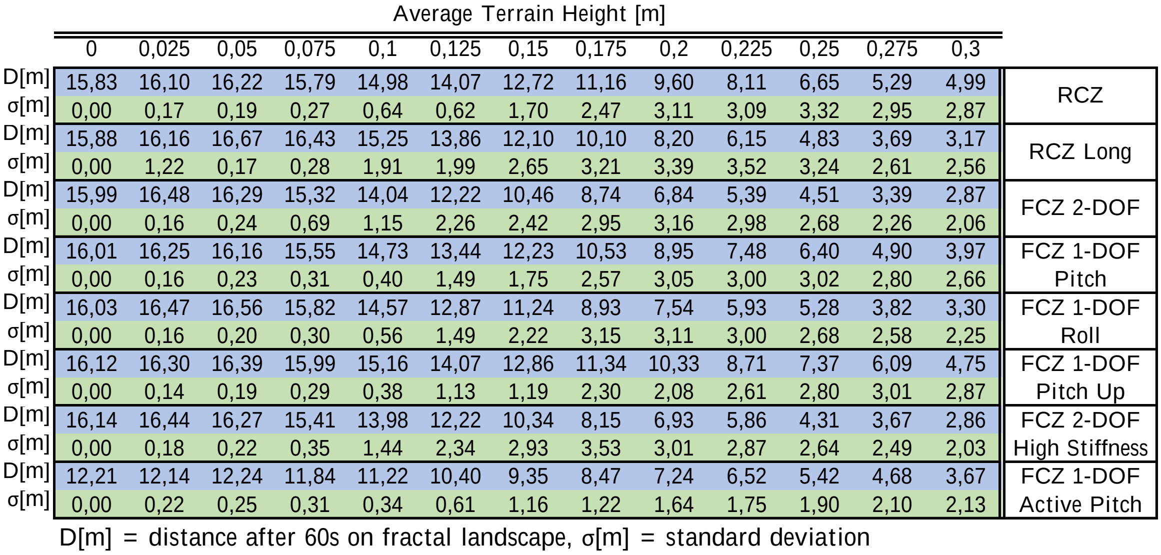


Figure A-20: Comparison between the 8 different 8-legged MCZ versions.

Comparison Between 10-Legged MCZ Robots



	Average Terrain Height [m]													
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	
D[m]	15,83	16,10	16,22	15,79	14,98	14,07	12,72	11,16	9,60	8,11	6,65	5,29	4,99	RCZ
σ[m]	0,00	0,17	0,19	0,27	0,64	0,62	1,70	2,47	3,11	3,09	3,32	2,95	2,87	
D[m]	15,88	16,16	16,67	16,43	15,25	13,86	12,10	10,10	8,20	6,15	4,83	3,69	3,17	RCZ Long
σ[m]	0,00	1,22	0,17	0,28	1,91	1,99	2,65	3,21	3,39	3,52	3,24	2,61	2,56	
D[m]	15,99	16,48	16,29	15,32	14,04	12,22	10,46	8,74	6,84	5,39	4,51	3,39	2,87	FCZ 2-DOF
σ[m]	0,00	0,16	0,24	0,69	1,15	2,26	2,42	2,95	3,16	2,98	2,68	2,26	2,06	
D[m]	16,01	16,25	16,16	15,55	14,73	13,44	12,23	10,53	8,95	7,48	6,40	4,90	3,97	FCZ 1-DOF Pitch
σ[m]	0,00	0,16	0,23	0,31	0,40	1,49	1,75	2,57	3,05	3,00	3,02	2,80	2,66	
D[m]	16,03	16,47	16,56	15,82	14,57	12,87	11,24	8,93	7,54	5,93	5,28	3,82	3,30	FCZ 1-DOF Roll
σ[m]	0,00	0,16	0,20	0,30	0,56	1,49	2,22	3,15	3,11	3,00	2,68	2,58	2,25	
D[m]	16,12	16,30	16,39	15,99	15,16	14,07	12,86	11,34	10,33	8,71	7,37	6,09	4,75	FCZ 1-DOF Pitch Up
σ[m]	0,00	0,14	0,19	0,29	0,38	1,13	1,19	2,30	2,08	2,61	2,80	3,01	2,87	
D[m]	16,14	16,44	16,27	15,41	13,98	12,22	10,34	8,15	6,93	5,86	4,31	3,67	2,86	FCZ 2-DOF High Stiffness
σ[m]	0,00	0,18	0,22	0,35	1,44	2,34	2,93	3,53	3,01	2,87	2,64	2,49	2,03	
D[m]	12,21	12,14	12,24	11,84	11,22	10,40	9,35	8,47	7,24	6,52	5,42	4,68	3,67	FCZ 1-DOF Active Pitch
σ[m]	0,00	0,22	0,25	0,31	0,34	0,61	1,16	1,22	1,64	1,75	1,90	2,10	2,13	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-21: Comparison between the 8 different 10-legged MCZ versions.

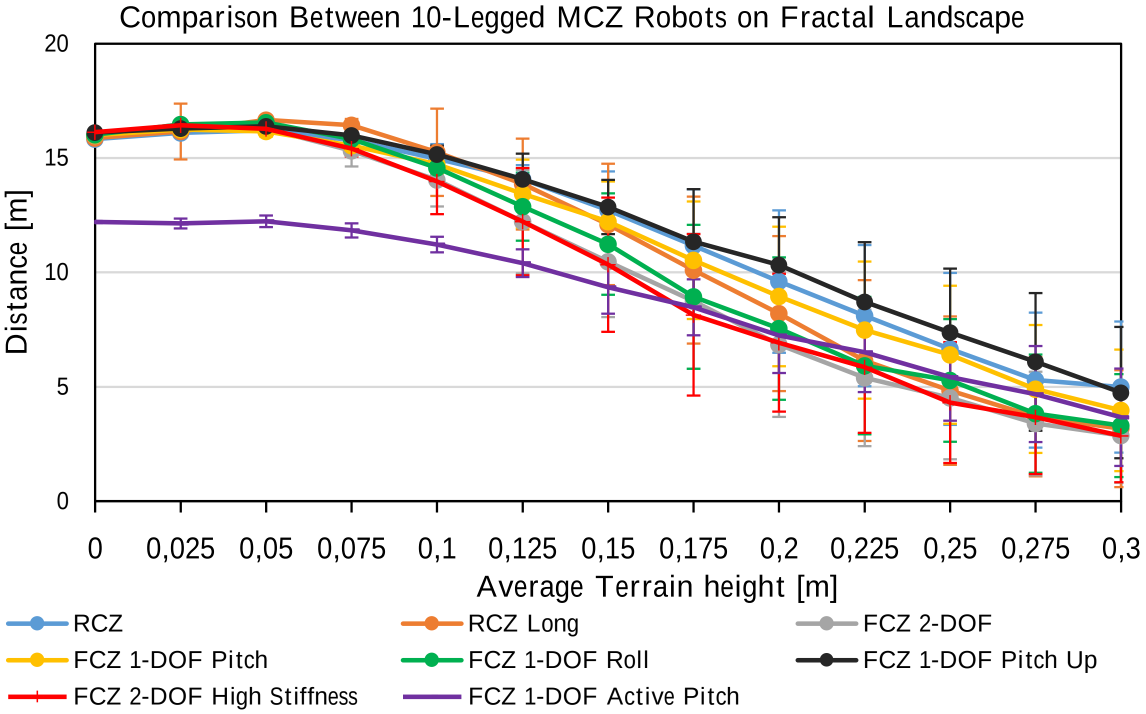


Figure A-22: Comparison between the 8 different 10-legged MCZ versions.

Comparison Between 12-Legged MCZ Robots

	Average Terrain Height [m]													
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	
D[m]	15,83	16,07	16,18	15,93	15,43	14,59	13,82	12,81	11,48	10,25	9,11	7,45	6,82	RCZ
σ[m]	0,00	0,15	0,22	0,27	0,38	1,19	1,07	1,54	2,24	2,47	2,82	3,08	3,13	
D[m]	15,86	16,18	16,64	16,52	15,95	14,84	13,35	12,32	10,46	8,78	8,00	6,33	5,34	RCZ Long
σ[m]	0,00	0,12	0,18	1,20	0,47	1,49	2,19	1,95	2,95	3,32	3,14	3,15	3,05	
D[m]	16,16	16,48	16,48	15,79	14,63	13,32	11,69	9,95	8,57	6,83	5,96	4,46	3,86	FCZ 2-DOF
σ[m]	0,00	0,19	0,19	0,34	0,61	1,28	1,87	2,61	2,92	3,01	2,84	2,56	2,58	
D[m]	16,17	16,33	16,34	15,84	14,62	13,15	11,41	9,94	8,39	6,90	6,34	4,78	4,71	FCZ 1-DOF Pitch
σ[m]	0,00	0,15	0,21	0,29	1,54	2,24	2,66	3,00	3,33	3,25	2,86	2,77	2,62	
D[m]	16,19	16,56	16,52	15,90	14,75	13,34	11,83	10,08	8,59	7,03	6,19	4,45	3,92	FCZ 1-DOF Roll
σ[m]	0,00	0,15	0,19	0,29	0,63	1,32	1,45	2,39	2,68	2,81	2,75	2,62	2,58	
D[m]	16,15	16,37	16,48	16,24	15,54	14,73	13,51	12,22	11,37	10,16	8,71	7,07	6,27	FCZ 1-DOF Pitch Up
σ[m]	0,00	0,14	0,19	0,27	0,39	0,48	1,20	1,93	1,88	1,95	2,53	3,01	3,00	
D[m]	16,14	16,48	16,43	15,79	14,74	13,34	11,62	10,18	8,49	7,58	5,58	4,89	3,88	FCZ 2-DOF High Stiffness
σ[m]	0,00	0,18	0,20	0,29	0,49	1,33	2,14	2,47	2,71	2,62	2,86	2,87	2,41	
D[m]	12,23	12,11	12,25	12,05	11,54	10,85	9,98	9,16	8,19	7,48	6,50	5,81	4,76	FCZ 1-DOF Active Pitch
σ[m]	0,00	0,19	0,24	0,27	0,35	0,50	0,73	1,10	1,48	1,51	1,85	2,00	2,19	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-23: Comparison between the 8 different 12-legged MCZ versions.

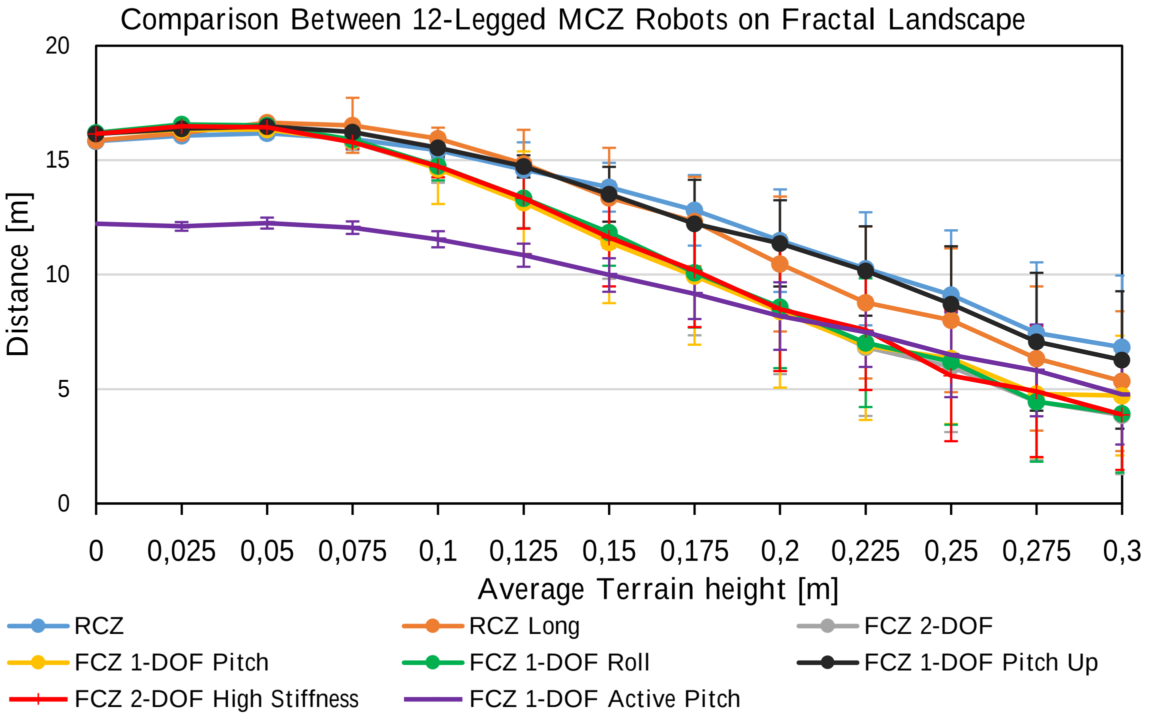


Figure A-24: Comparison between the 8 different 12-legged MCZ versions.

Comparison Between 14-Legged MCZ Robots

	Average Terrain Height [m]													
	0	0,025	0,05	0,075	0,1	0,125	0,15	0,175	0,2	0,225	0,25	0,275	0,3	
D[m]	15,87	16,00	16,22	16,05	15,61	15,06	14,31	13,33	12,33	11,33	10,24	8,89	7,48	RCZ
σ[m]	0,00	1,15	0,22	0,26	0,33	0,51	1,08	1,78	2,19	2,03	2,51	2,89	3,33	
D[m]	15,89	16,09	16,61	16,71	16,21	15,38	14,28	12,75	11,00	9,77	8,07	7,08	6,10	RCZ Long
σ[m]	0,00	1,15	0,17	0,26	0,40	0,54	1,38	2,23	3,17	3,40	3,44	3,60	3,46	
D[m]	16,35	16,43	16,23	15,48	14,36	12,75	11,39	9,73	8,10	6,98	6,06	4,65	4,30	FCZ 2-DOF
σ[m]	0,00	0,17	0,21	0,31	0,55	1,64	1,85	2,44	3,06	2,72	2,77	2,59	2,55	
D[m]	16,11	16,22	16,26	15,71	14,63	13,08	11,51	9,95	7,92	6,64	5,89	4,90	4,09	FCZ 1-DOF Pitch
σ[m]	0,00	0,15	0,21	0,33	0,73	1,57	2,41	2,67	3,36	3,01	3,04	2,71	2,57	
D[m]	16,25	16,64	16,53	15,98	14,90	13,82	12,32	10,80	9,38	7,51	6,99	5,47	4,89	FCZ 1-DOF Roll
σ[m]	0,00	0,16	0,23	0,30	0,72	0,84	1,71	2,34	2,60	2,84	2,75	2,56	2,53	
D[m]	16,16	16,33	16,45	16,22	15,58	14,64	13,75	12,38	11,24	9,52	8,79	7,72	6,70	FCZ 1-DOF Pitch Up
σ[m]	0,00	0,14	0,19	0,26	0,43	1,24	1,20	1,79	2,13	2,85	2,79	2,74	3,01	
D[m]	16,29	16,44	16,21	15,52	14,38	12,94	11,38	9,78	8,32	7,05	5,79	4,70	4,33	FCZ 2-DOF High Stiffness
σ[m]	0,00	0,17	0,23	0,33	0,91	1,20	2,11	2,42	2,83	2,79	2,62	2,73	2,51	
D[m]	12,34	12,19	12,33	12,14	11,71	11,07	10,29	9,49	8,63	7,77	6,84	6,05	5,20	FCZ 1-DOF Active Pitch
σ[m]	0,00	0,21	0,24	0,28	0,36	0,55	0,76	0,98	1,17	1,66	1,92	2,15	2,02	

D[m] = distance after 60s on fractal landscape, σ[m] = standard deviation

Figure A-25: Comparison between the 8 different 14-legged MCZ versions.

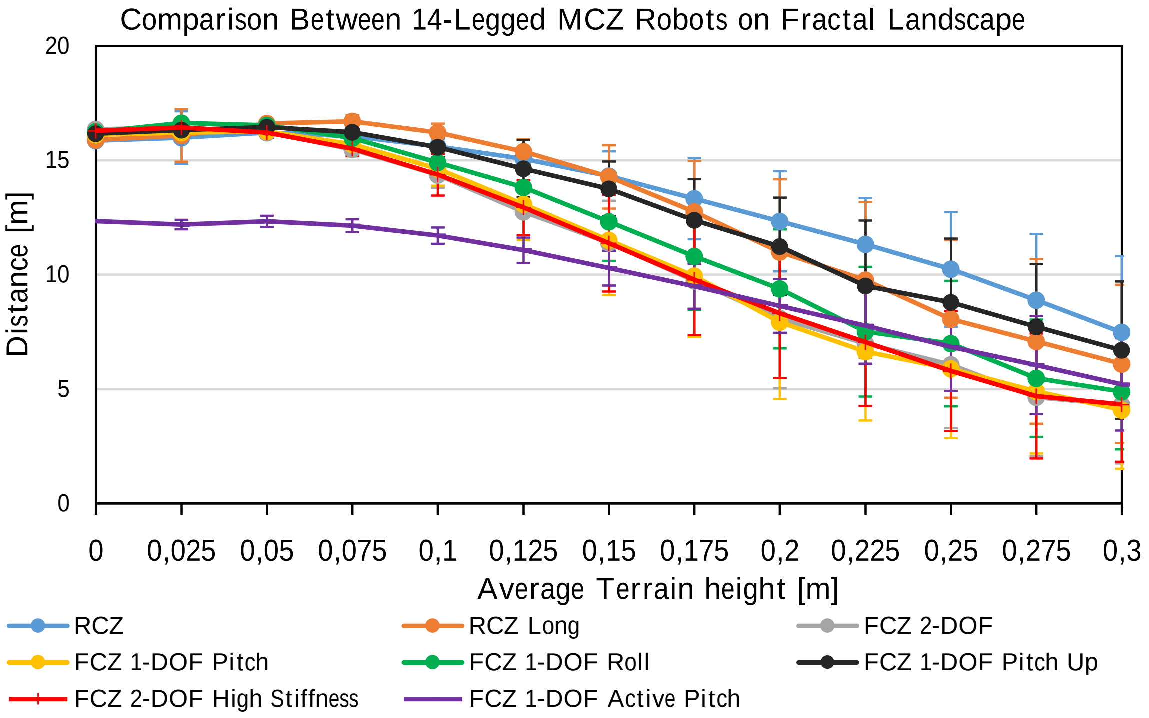
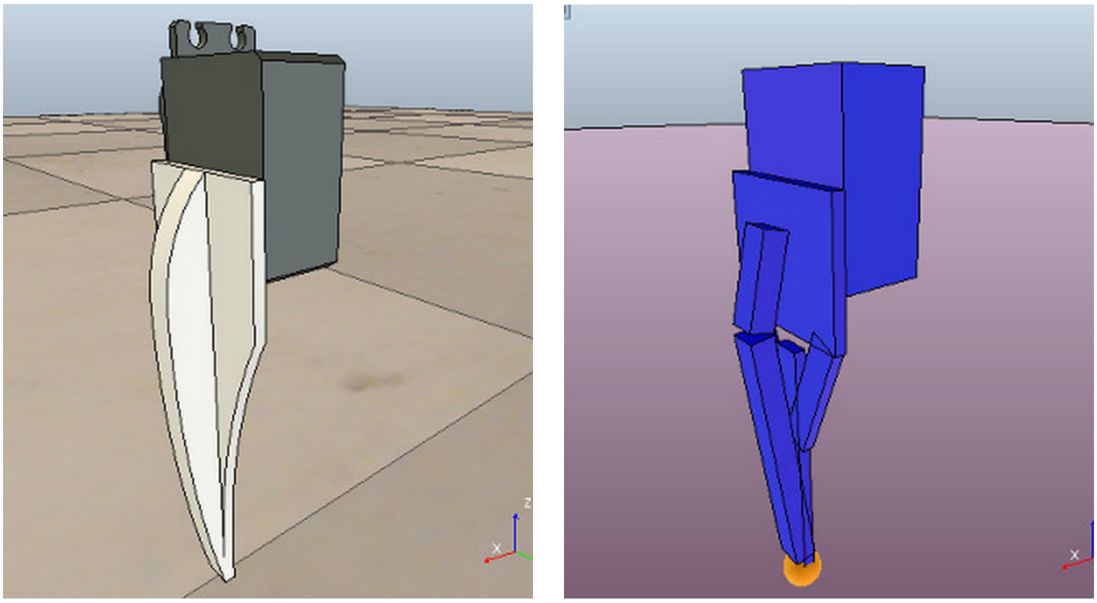


Figure A-26: Comparison between the 8 different 14-legged MCZ versions.

Appendix B

Making models in V-REP

To run a simulation in V-REP, different elements are necessary. First a scene must be loaded, which is where the simulation takes place. A scene consists of an environment (standard is an empty space with a flat ground surface), a main script which controls the simulation, and several objects like the camera, lights, and the floor. For the simulations in this thesis, only the standard scene with an enlarged floor will be used. Obstacles will be put in as objects. A model is a sub-element of a scene. It consist of a certain shape and can contain a number of joints and sensors. The shapes can either be created in V-REP, or imported from a Computer-Aided Design (CAD) program. V-REP uses two different kinds of shapes; pure shapes and non-pure shapes. Pure shapes are primitive shapes (a plane, disc, cuboid, sphere or cylinder), and they are used for the dynamics calculations. They can be assigned with a certain mass and some other material properties. Non-pure shapes don't have to be primitive shapes, they can have any possible shape necessary. Non-pure shapes are used for collision detection, distance calculation, object detection, and to render the graphics. In order for a model to move in a simulation, it must consist of pure shapes (for the dynamics), and most of the time it also contains non-pure shapes for the visual rendering and other functions like collision detection, etc. When a model is imported from a mesh file, it loads as one whole non-pure shape. Often, the original model consist of multiple parts, which means that they have to be extracted from the non-pure shape. V-REP has a build-in function to extract these parts automatically. When all non-pure shapes are separated, a pure shaped model must be created for the dynamics calculations. In V-REP there exists a mode called triangle edit mode, in which all the triangles that make up a shape are displayed. When a combination of triangles is selected, a primitive shape can be fitted to match the selected triangles as well as possible. In this way, the pure shapes which make up the dynamics model can be created. Sometimes, a non-pure shape is too complicated to fit a primitive shape to. In that case, a pure shape has to be built manually by combining multiple primitive shapes and placing them to match the non-pure shape. How this is done can be seen in figure B-1, where a pure shape is manually made for a hexapod leg by placing primitive shapes in a similar configuration. After the pure-shaped model is completed, all of the moving parts are connected by joints to the model base. This is done with a parent-child system, where the model base is parent of the joint, which is in its turn parent of the moving part. Revolute, prismatic and spherical joints are available in the standard V-REP library. Sensors can also be added in the same way as joints. By using this process, a complete robot model can be built relatively easy. When the model is finished, it can be controlled by using a child script. This is a script attached to the robot, which contains Lua code with Application Programming Interface (API) functions, so all the sensors can be read out, as well as many different simulation parameters. Values can be send to the joints, which have their own build-in PID-controllers to control the position. When the robot model is finished, it can be saved so it can later be used to run simulations in a certain scene.



(a) Non-pure shape of a hexapod leg.

(b) Pure shape of a hexapod leg.

Figure B-1: A manually made pure shape for a hexapod leg.

Appendix C

Gait Generation

For the robot to be able to move, several controllers are working together to calculate the necessary control signals. One of these is a max-plus gait scheduling system that synchronizes the legs and makes it possible to change gaits smoothly during movement [31] [32]. The algebra used for gait generation and switching is explained in this appendix. More information on the gaits used for different scenarios can be found in chapter 3.

C-1 Max-Plus Algebra

In engineering, linear algebra is one of the most used mathematical tools. The traditional algebra which is used primarily, has the structure of a ring:

$$(\mathbb{R}, +, \times, 0, 1)$$

where:

$\mathbb{R}$	it the set of real numbers
$+$	is the plus operation
$\times$	is the multiplication operation
0	is the absorbing element
1	is the identity element

Traditional algebra has the following properties:

- It is an abelian group under addition. To qualify as an abelian group, the set and operation $(\mathbb{R}, +)$ must satisfy five requirements known as the abelian group axioms:
  - **Closure**: For all a, b in $\mathbb{R}$, the result of the operation $a+b$ must also be in $\mathbb{R}$. This satisfies since adding any set of real numbers results in a real number.
  - **Associativity**: For all a, b and c in $\mathbb{R}$, the equation $(a+b)+c = a+(b+c)$ holds.
  - **Identity element**: There exists an element e in $\mathbb{R}$, such that for all elements a in $\mathbb{R}$, the equation $e + a = a + e = a$ holds. For the addition operation this identity element is 0.
  - **Inverse element** For each a in $\mathbb{R}$, there exists an element b in $\mathbb{R}$ such that $a+b = b + a = e$, where e is the identity element. For the addition operation the inverse element of a would be $-a$.
  - **Commutativity**: For all a, b in $\mathbb{R}$, $a + b = b + a$.
- It is a monoid under multiplication. To qualify as a monoid group, the set and operation $(\mathbb{R}, \times)$ must satisfy two requirements:
  - **Associativity**: For all a, b and c in $\mathbb{R}$, the equation $(a \times b) \times c = a \times (b \times c)$ holds.
  - **Identity element**: There exists an element e in $\mathbb{R}$, such that for all elements a in $\mathbb{R}$, the equation $e \times a = a \times e = a$ holds. For the multiplication operation this identity element is 1.
- Multiplication is distributive over addition: for any numbers a, b and c the following two equations hold:

$$\begin{aligned} a \times (b + c) &= (a \times b + a \times c) \\ (b + c) \times a &= (b \times a + c \times a) \end{aligned}$$

It is possible to define arbitrary algebras as long as they satisfy the ring axioms. Max-plus algebra is considered a tropical (exotic) algebra that uses different sets and operators from the traditional algebra. It has the following structure:

$$(\mathbb{R}_{max}, \oplus, \otimes, \varepsilon, e)$$

$\mathbb{R}_{max} := \mathbb{R} \cup \{-\infty\}$	the set is the reals $(\mathbb{R})$ with the extra element minus infinity $(-\infty)$.
$x \oplus y := max(x, y)$	the plus operation.
$x \otimes y := x + y$	the times operation.
$\varepsilon := -\infty$	the zero element
$e := 0$	the identity element

The plus operation is the maximum of the provided elements and it has no inverse (once the maximum is taken there is no way to get back to the original elements). The times operation is the addition of the provided elements. Max-plus algebra has the structure of a commutative idempotent semi-ring. It is **commutative** since $a \oplus b = b \oplus a$ and $a \otimes b = b \otimes a$, and it is **idempotent** since the $\oplus$ operation can be repeated multiple times and will always give the same result: $a \oplus b = c, a \oplus c = c, b \oplus c = c$, etc. Furthermore it is a **semi-ring** since there is no inverse for the plus operation. The following examples will demonstrate clearly how the operators for max-plus algebra work:

- The plus and times operations:

$$\begin{aligned} 3 \oplus 5 &= max(3,5) = 5 \\ 1 \otimes 2 &= 1 + 2 = 3 \end{aligned}$$

- The absorbing element:

$$\begin{aligned} 3 \oplus \varepsilon &= max(3,-\infty) = 3 \\ 3 \otimes \varepsilon &= 5 - \infty = -\infty = \varepsilon \end{aligned}$$

- The identity element:

$$\begin{aligned} 3 \oplus e &= max(3,0) = 3 \\ 3 \otimes e &= 3 + 0 = 3 \end{aligned}$$

The max-plus algebra can also be expanded to matrices:

$$(\mathbb{R}_{max}^{n\times m}, \oplus, \otimes, \mathcal{E}, E)$$

with

$$\begin{aligned} [A \oplus B]_{ij} =& a_{ij} \oplus b_{ij} := max(a_{ij}, b_{ij}) \\ [A \oplus C]_{ij} =& \bigoplus_{k=1}^{m} a_{ik} \oplus c_{kj} := \max_{k=1,\ldots,m} (a_{ik} + c_{kj}) \end{aligned}$$

The identity and zero matrices are defined by:

$$\begin{aligned} [\mathcal{E}]_{ij} &= \varepsilon \\ [E]_{ij} &= \begin{cases} e & \text{if } i = j \\ \varepsilon & \text{otherwise} \end{cases} \end{aligned}$$

The k-th power of a matrix is defined by

$$D^{\otimes k} := \underbrace{D \otimes D \otimes \cdots \otimes D}_{k-\text{times}}$$

C-1-1 Max-plus algebra for locomotion

Max-plus algebra is used for many applications where time scheduling and synchronization is useful. This section explains how it can be used for gait scheduling in legged robots. When a robot is walking, each leg is either on the ground (stance phase), or in the air (swing phase). The time it spends in the air after lift-off until the next touchdown is called the flight time (τ_f), and the time it spends on the ground after touchdown until the next lift-off is called the ground time (τ_g). The time that the legs spend on the ground together is called the double-stance time (τ_Δ). The following parameters are declared:

$t_i(k)$ is the touchdown time for leg i at event iteration k
$l_i(k)$ is the lift-off time for leg i at event iteration k
τ_f is the flight time (swing)
τ_g is the ground time (stance)
τ_Δ is the double stance time

For a robot with 2 legs, the following equations describe the leg cycles:

$$\begin{aligned} t_1(k+1) &= l_1(k+1) + \tau_f && \text{(C-1)} \\ l_1(k+1) &= t_1(k) + \tau_g && \text{(C-2)} \\ t_2(k+1) &= l_2(k+1) + \tau_f && \text{(C-3)} \\ l_2(k+1) &= t_2(k) + \tau_g && \text{(C-4)} \end{aligned}$$

These equations describe the lift-off and touchdown times for the two legs. The next touchdown time of leg 1 ($t_1(k+1)$) is its lift-off time ($l_1(k+1)$) plus the time it spends in the air (τ_f), as can be seen in equation C-1. The next lift-off time of leg 1 ($l_1(k+1)$) is its touchdown time ($t_1(k)$) plus the time it spends on the ground (τ_g) (equation C-2). The same can be done for leg 2 (equation C-3 and C-4).

Next, the legs need to be synchronized with respect to each other while considering the double stance time. This means that leg 1 can only lift off after leg 2 has touched down and they have been together on the ground during the double stance time. For this condition, lift-off can happen at $t_2(k) + \tau_\Delta$. The other condition is that leg 1 has to complete its own ground time. This is why the maximum is taken of these two time instances (eq. C-6). There is no special condition for touchdown, so this equation remains the same (eq. C-5). The same is then done for the other leg (eq. C-7 and C-8).

$$
\begin{aligned}
t_1(k+1) &= l_1(k+1) + \tau_f &\text{(C-5)}\\
l_1(k+1) &= max(t_1(k) + \tau_g, t_2(k) + \tau_\Delta) &\text{(C-6)}\\
t_2(k+1) &= l_2(k+1) + \tau_f &\text{(C-7)}\\
l_2(k+1) &= max(t_2(k) + \tau_g, t_1(k+1) + \tau_\Delta) &\text{(C-8)}
\end{aligned}
$$

By substituting the equations to get rid of the $(k+1)$ instances on the right hand side and rewriting them in max-plus algebra form, the following equation is obtained which is a max-plus linear system:

$$
\underbrace{\begin{bmatrix} t_1(k+1) \\ t_2(k+1) \\ l_1(k+1) \\ l_2(k+1) \end{bmatrix}}_{x(k+1)} = \underbrace{\begin{bmatrix} \tau_f \otimes \tau_g & \tau_f \otimes \tau_\Delta & \varepsilon & \varepsilon \\ \tau_f^{\otimes 2} \otimes \tau_g \otimes \tau_\Delta & (\tau_f \otimes \tau_\Delta)^{\otimes 2} & \varepsilon & \varepsilon \\ \tau_g & \tau_\Delta & \varepsilon & \varepsilon \\ \tau_f \otimes \tau_g \otimes \tau_\Delta & \tau_f \otimes \tau_\Delta^{\otimes 2} & \varepsilon & \varepsilon \end{bmatrix}}_{A} \otimes \underbrace{\begin{bmatrix} t_1(k) \\ t_2(k) \\ l_1(k) \\ l_2(k) \end{bmatrix}}_{x(k)} \tag{C-9}
$$

For an n-legged robot, the following full discrete event state vector can be defined:

$$
x(k) = [\underbrace{t_1(k) \dots t_n(k)}_{t(k)} \quad \underbrace{l_1(k) \dots l_n(k)}_{l(k)}]^T
$$

This vector contains the touchdown times for each leg for iteration k in the first half of its columns, and the lift-off times in the second half. For a synchronized gait one can write the modelling equation as follows:

$$
\begin{bmatrix} t(k+1) \\ l(k+1) \end{bmatrix} = \left[\begin{array}{c|c} \mathcal{E} & \tau_f \otimes E \\ \hline P & \mathcal{E} \end{array}\right] \otimes \begin{bmatrix} t(k+1) \\ l(k+1) \end{bmatrix} \oplus \left[\begin{array}{c|c} E & \mathcal{E} \\ \hline \tau_g \otimes E \oplus Q & E \end{array}\right] \otimes \begin{bmatrix} t(k) \\ l(k) \end{bmatrix} \tag{C-10}
$$

When these matrix multiplications are written out, the results are two familiar equations. The first equation says that touchdown occurs τ_f seconds after lift-off, which is similar to equations C-5 and C-7:

$$
t(k+1) = \tau_f \otimes E \otimes l(k+1) \oplus t(k) \tag{C-11}
$$

One difference with equations C-5 and C-7 is the $\oplus t(k)$ on the right side which ensures that the next touchdown time is larger then the previous touchdown time of the same leg. The second equation says lift-off occurs τ_g seconds after touchdown and after synchronization, which is similar to equations C-6 and C-8:

$$
l(k+1) = P \otimes t(k+1) \oplus (\tau_g \otimes E \oplus Q) \otimes t(k) \oplus l(k) \tag{C-12}
$$

The P and Q matrices determine which legs should synchronize together depending on the gait. How these are built will be explained later in this section. The $\oplus l(k)$ on the right side ensures that the next lift-off can not happen before the previous lift-off of the same leg. Equation C-10 is an **implicit** expression, but it is possible to rewrite it to an equivalent **explicit** max-plus linear system. Consider the max-plus linear system:

$$x = A \otimes x \oplus b \tag{C-13}$$

Define the matrix A^* as

$$A^* := \bigoplus_{k=0}^{\infty} A^{\otimes k} \tag{C-14}$$

If A^* exists, then

$$x = A^* \otimes b \tag{C-15}$$

solves the system.[1] For gait scheduling, the following matrix structure is used (see eq. C-10):

$$x(k+1) = A_0 \otimes x(k+1) \oplus A_1 \otimes x(k) \tag{C-16}$$

First the matrix A_0^* is computed:

$$A_0^* := \bigoplus_{k=0}^{\infty} A_0^{\otimes k} \tag{C-17}$$

If A_0^* exists, then the system can be written in an explicit form:

$$\begin{aligned} x(k+1) &= A_0^* \otimes A_1 \otimes x(k) && \text{(C-18)} \\ &= A \otimes x(k) && \text{(C-19)} \end{aligned}$$

with

$$A = A_0^* \otimes A_1$$

Generating gaits The gaits can be designed with the use of the P and Q matrices in equation C-10. First, the following notation is introduced: let $l_1, \ldots, l_m$ be sets of integers such that

$$\bigcup_{p=1}^{m} l_p = \{1, \ldots, n\}, \text{and}$$
$$\forall i \neq j, l_i \cap l_j = \emptyset$$

[1] Baccelli 1992, Lemma 2.2

This means that every set of integers is different and each integer can only be used in one set. A gait $\mathcal{G}$ is defined as an ordering relation of groups of legs:

$$\mathcal{G} = l_1 \prec l_2 \prec \ldots \prec \text{ł}_m$$

Each set of integers represents a group of legs that move together. A gait is formed by subsequent sets of integers. An example of two gaits is given to clarify this concept. Suppose that the leg movement of a basic hexapod robot needs to be scheduled. When it is moving in a tripod gait, three legs remain on the ground while three other legs are in flight phase, as can be seen in figure C-1. First legs 1, 4 and 5 move in flight phase while 2, 3 and 6 remain in stance. When the first set of legs touch down, the second set of legs goes into flight phase. This corresponds to gait $\mathcal{G}_{tripod} = \{1, 4, 5\} \prec \{2, 3, 6\}$. A second example is given with a quadruped gait, as can be seen in figure C-1. With this gait, 4 of the 6 legs remain in stance at the same time. This corresponds to gait $\mathcal{G}_{quadruped} = \{1, 4\} \prec \{3, 6\} \prec \{2, 5\}$. The legs of the Modular Centipede Zebro (MCZ) robot, which is described in more detail in chapter 2, are numbered in the same way as with the Zebro robot.

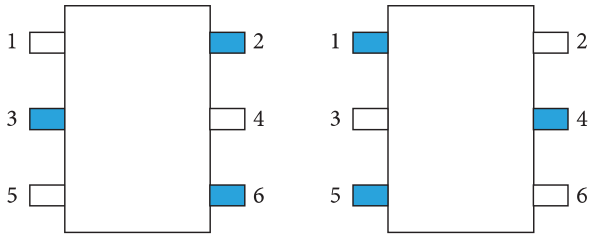


Figure C-1: Example of a tripod gait on a hexapod robot.

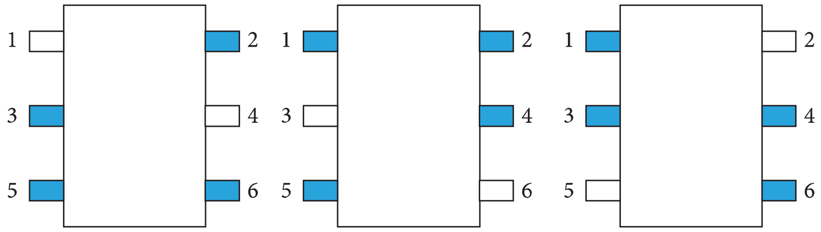


Figure C-2: Example of a quadruped gait on a hexapod robot.

The P and Q matrices can be constructed automatically for every possible gait $\mathcal{G}$ with the following definition:

For

$$\forall j \in \{1, \ldots, m-1\}$$
$$\forall p \in l_{j+1}$$
$$\forall q \in l_j$$

let

$$[P]_{p,q} = \tau_\Delta$$

and for

$$\forall p \in l_1$$
$$\forall q \in l_m$$

let

$$[Q]_{p,q} = \tau_\Delta$$

All other entries of P and Q are ε. As an example, the gait $\mathcal{G} = \{1,4\} \prec \{2,3\}$ will result in the following P and Q matrices:

$$P = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \tau_\Delta & \varepsilon & \varepsilon & \tau_\Delta \\ \tau_\Delta & \varepsilon & \varepsilon & \tau_\Delta \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} \quad Q = \begin{bmatrix} \varepsilon & \tau_\Delta & \tau_\Delta & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \tau_\Delta & \tau_\Delta & \varepsilon \end{bmatrix}$$

By plugging in the P en Q matrices, equation C-18 can be used to calculate the next iteration in the touchdown and lift-off times. Repeating this process will result in an event list with all the touchdown and lift-off times. As an example, using the parameter set $\tau_f = 1s, \tau_g = 1.4s$, and $\tau_\Delta = 0.2s$ will result in the following event list:

k	$t_1(k)$	$t_2(k)$	$t_3(k)$	$t_4(k)$	$l_1(k)$	$l_2(k)$	$l_3(k)$	$l_4(k)$
0	0	0	0	0	0	0	0	0
1	2.4	3.6	3.6	2.4	1.4	2.6	2.6	1.4
2	4.8	6.0	6.0	4.8	3.8	5.0	5.0	3.8
3	7.2	8.4	8.4	7.2	6.2	7.4	7.4	6.2
4	9.6	10.8	10.8	9.6	8.6	9.8	9.8	8.6
5	12.0	13.2	13.2	12.0	11.0	12.2	12.2	11.0

Table C-1: Event list with touchdown and lift-off times for 5 iterations with the parameter set $\tau_f = 1s, \tau_g = 1.4s$, and $\tau_\Delta = 0.2s$ and gait $\mathcal{G} = \{1,4\} \prec \{2,3\}$

One of the advantages of using max-plus algebra to calculate these time instances, is that for each iteration a different gait can be used. Simply using different P and Q matrices in equation C-18 at the next iteration step will smoothly synchronize the legs to a new gait. The last step is to convert the event times to a reference signal for the legs. This is done by linear interpolation between the lift-off angle θ_l (where the leg will go into flight phase) and the touchdown angle θ_t (where the leg will touch the ground) considering the event times. A function called the **continuous time scheduler** will convert the discrete events into desired continuous time trajectories with the use of the following algorithm:

$$\theta_{ref,i}(\tau) := \begin{cases} \dfrac{\theta_l(t_i(k_{2i-1}) - \tau) + (\theta_t + 2\pi)(\tau - l_i(k_{2i}))}{t_i(k_{2i-1}) - l_i(k_{2i})} & \text{if } \tau \in [l_i(k_{2i}), t_i(k_{2i-1})) \\ \dfrac{\theta_t(l_i(k_{2i}+1) - \tau) + \theta_l(\tau - t_i(k_{2i-1}))}{l_i(k_{2i}+1) - t_i(k_{2i-1})} & \text{if } \tau \in [t_i(k_{2i-1}), l_i(k_{2i}+1)) \end{cases} \tag{C-20}$$

where θ_l and θ_t are the lift-off and touchdown angles respectively, and τ is the current time instant. These continuous time trajectories are send to the individual PID-controllers in the leg joints, enabling the robot to move. To summarize, the robot is controlled by a supervisory control function that picks the gait and calculates the P, Q, and A matrices with the use of a parameter set $\tau_f, \tau_g, \tau_\Delta, \theta_l$, and θ_t. The A matrix is passed on to the max-plus gait scheduler (equation C-18), which calculates the event list with the lift-off and touchdown times t_i and l_i. Then, the continuous time scheduler converts this into a reference signal $\theta_{ref,i}$ for each leg, which is tracked by the PID-controllers in the joints to enable the robot to move (figure C-3).

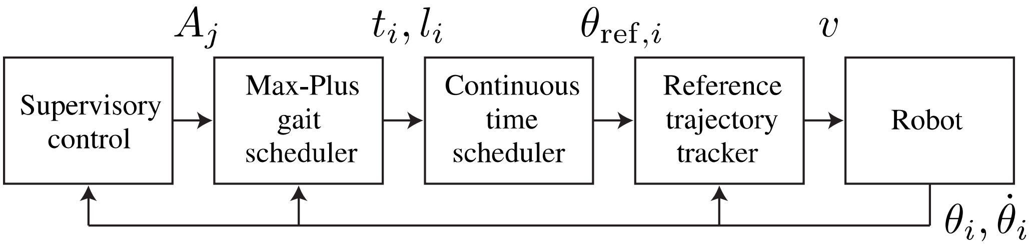


Figure C-3: Control structure of a legged robot.

Bibliography

[1] R. Rescue, “Robocup rescue website,” Jan. 2006.

[2] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, *et al.*, “Bigdog, the rough-terrain quadruped robot,” in *Proceedings of the 17th World Congress*, vol. 17, pp. 10822–10825, 2008.

[3] U. Saranli, M. Buehler, and D. E. Koditschek, “Rhex: A simple and highly mobile hexapod robot,” *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001.

[4] K. C. Galloway, G. C. Haynes, B. D. Ilhan, A. M. Johnson, R. Knopf, G. A. Lynch, B. N. Plotnick, M. White, and D. E. Koditschek, “X-rhex: A highly mobile hexapedal robot for sensorimotor tasks,” 2010.

[5] G. C. Haynes, J. Pusey, R. Knopf, A. M. Johnson, and D. E. Koditschek, “Laboratory on legs: an architecture for adjustable morphology with legged robots,” in *SPIE Defense, Security, and Sensing*, pp. 83870W–83870W, International Society for Optics and Photonics, 2012.

[6] K. C. Galloway, J. E. Clark, M. Yim, and D. E. Koditschek, “Experimental investigations into the role of passive variable compliant legs for dynamic robotic locomotion,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1243–1249, IEEE, 2011.

[7] D. Campbell and M. Buehler, “Stair descent in the simple hexapod’rhex’,” in *Robotics and Automation, 2003. Proceedings. ICRA’03. IEEE International Conference on*, vol. 1, pp. 1380–1385, IEEE, 2003.

[8] E. Moore, D. Campbell, F. Grimminger, and M. Buehler, “Reliable stair climbing in the simple hexapod’rhex’,” in *Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on*, vol. 3, pp. 2222–2227, IEEE, 2002.

[9] A. M. Johnson and D. Koditschek, “Toward a vocabulary of legged leaping,” in *IEEE Int. Conf. Rob. Aut. pp.(to appear)*, 2013.

[10] M. Khoramshahi, A. Sprowitz, A. Tuleu, M. Ahmadabadi, and A. Ijspeert, “Benefits of an active spine supported bounding locomotion with a small compliant quadruped robot,” 2013.

[11] M. H. H. Kani, M. Derafshian, H. J. Bidgoly, and M. N. Ahmadabadi, “Effect of flexible spine on stability of a passive quadruped robot: Experimental results,” in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pp. 2793–2798, IEEE, 2011.

[12] G. A. Folkertsma, S. Kim, and S. Stramigioli, “Parallel stiffness in a bounding quadruped with flexible spine,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2210–2215, IEEE, 2012.

[13] S. Seok, A. Wang, M. Y. Chuah, D. Otten, J. Lang, and S. Kim, “Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 3307–3312, IEEE, 2013.

[14] R. T. Schroer, M. J. Boggess, R. J. Bachmann, R. D. Quinn, and R. E. Ritzmann, “Comparing cockroach and whegs robot body motions,” in *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, vol. 4, pp. 3288–3293, IEEE, 2004.

[15] M. Eich, F. Grimminger, S. Bosse, D. Spenneberg, and F. Kirchner, “Asguard: A hybrid-wheel security and sar-robot using bio-inspired locomotion for rough terrain,” *Proceedings ROBIO 2008*, pp. 774–779, 2008.

[16] D. Koh, J. Yang, and S. Kim, “Centipede robot for uneven terrain exploration: Design and experiment of the flexible biomimetic robot mechanism,” in *Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on*, pp. 877–881, IEEE, 2010.

[17] F. Enner, D. Rollinson, and H. Choset, “Simplified motion modeling for snake robots,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 4216–4221, IEEE, 2012.

[18] S. Hirose and E. F. Fukushima, “Development of mobile robots for rescue operations,” *Advanced Robotics*, vol. 16, no. 6, pp. 509–512, 2002.

[19] K. Suzuki, A. Nakano, G. Endo, and S. Hirose, “Development of multi-wheeled snake-like rescue robots with active elastic trunk,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 4602–4607, IEEE, 2012.

[20] M. Yim, P. White, M. Park, and J. Sastra, “Modular self-reconfigurable robots,” in *Encyclopedia of complexity and systems science*, pp. 5618–5631, Springer, 2009.

[21] S. Murata, K. Kakomura, and H. Kurokawa, “Toward a scalable modular robotic system,” *Robotics & Automation Magazine, IEEE*, vol. 14, no. 4, pp. 56–63, 2007.

[22] J. Sastra, W. Bernal-Heredia, J. Clark, and M. Yim, “A biologically-inspired dynamic legged locomotion with a modular reconfigurable robot,” in *Proc. of DSCC ASME Dynamic Systems and Control Conference, Ann Arbor, Michigan, USA*, 2008.

[23] D. Miner, J. Glaros, and T. Oates, “Self-configuring modular centipede robot,” in *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing*, pp. 910–14, Citeseer, 2007.

[24] K. Jeong, J. Kang, G. Lee, S.-u. Lee, Y. Seo, S. Jung, and S. Kim, “A remotely operated robotic system for urban search and rescue,” in *SICE-ICASE, 2006. International Joint Conference*, pp. 3142–3145, IEEE, 2006.

[25] A. Torige, M. Noguchi, and N. Ishizawa, “Centipede type multi-legged walking robot,” in *Intelligent Robots and Systems’ 93, IROS’93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 1, pp. 567–571, IEEE, 1993.

[26] R.-T. P. Simulation, “Bullet physics library website,” Jan. 2014.

[27] R. Smith, “Open dynamics engine website,” Jan. 2000.

[28] C. Labs, “Vortex website,” Jan. 2014.

[29] PUC-Rio, “Lua website,” Jan. 2014.

[30] R. Altendorfer, U. Saranli, H. Komsuoglu, D. Koditschek, H. B. Brown Jr, M. Buehler, N. Moore, D. McMordie, and R. Full, *Evidence for spring loaded inverted pendulum running in a hexapod robot.* Springer, 2001.

[31] G. Lopes, B. Kersbergen, T. J. van den Boom, B. De Schutter, R. Babuska, *et al.*, “Modeling and control of legged locomotion via switching max-plus models,” *Robotics, IEEE Transactions on*, vol. 30, no. 3, pp. 652–665, 2014.

[32] G. A. Lopes, B. Kersbergen, B. De Schutter, T. van den Boom, and R. Babuška, “Synchronization of a class of cyclic discrete-event systems describing legged locomotion,” *Discrete Event Dynamic Systems*, pp. 1–37, 2011.

Glossary

List of Acronyms

TU Delft	Delft University of Technology
DOF	degree-of-freedom
DOFs	degrees-of-freedom
Tokio Tech	Tokyo Institute of Technology
LIDAR	Laser Imaging Detection And Ranging
DARPA	Defense Advanced Research Projects Agency
IMU	Inertial Measurement Unit
TU Delft	Delft University of Technology
NIST	U.S. National Institute of Standards and Technology
USAR	Urban Search and Rescue
MSR	Modular Self-Reconfigurable
SLIP	spring-loaded inverted pendulum
AUVs	Unmanned Areal Vehicles
API	Application Programming Interface
ROS	Robot Operating System
CAD	Computer-Aided Design
MCZ	Modular Centipede Zebro
FCZ	Flexible Centipede Zebro
RCZ	Rigid Centipede Zebro
ANOVA	Analysis of Variance